HZ BOOKS
华章IT

区块链
技术指南

邹 均 张海宁 唐 屹 李 磊 等著

权威区块链专家联袂推荐，资深区块链践行者联合撰写，从技术层面全面解密区块链技术

系统讲解区块链核心概念、架构、底层算法、应用开发、典型项目与应用、常见问题等读者最为关心的技术与应用

机械工业出版社
China Machine Press

内容简介

权威区块链专家联袂推荐，资深区块链践行者联合撰写，从技术层面全面解密区块链技术。涵盖基础概念、架构、底层算法、应用开发、典型的区块链解决方案、常见问题等读者最为关心的技术与应用。

本书分为三大部分，共计11章内容。

第一部分为基础和入门（第1～2章），着重区块链入门介绍，讲解区块链基础，包括区块链的概念、种类、比特币交易、区块链的一些基本概念等，为后面深入介绍区块链技术做铺垫。

第二部分为架构和技术篇（第3～10章）：详细讲解了以下方面。

- 区块链1.0/2.0/3.0架构，以及互联链架构属性与特点剖析，做到宏观理解与认识。
- 区块链基于的密码学原理和典型的算法，了解区块链开发安全之道。
- 区块链中常用的共识算法与作用，了解区块链价值传递与弱中心化之基石。
- 比特币开发指南，帮助初学者入门。
- 以太坊上的智能合约开发要领，为以后应用打下基础。
- HyperLedger开源项目及其架构，掌握主流的项目与应用。
- 区块链上常见的问题，包括The DAO攻击的源码级分析。
- 典型解决方案：以闪电网络为主的支付方案；以标识登记为主的开源ODIN解决方案。

第三部分为回顾和展望（第11章），从架构变革的角度探讨IT发展的原动力，并提供对区块链对未来IT发展的一些展望。

区块链
技术指南

邹均 张海宁 唐屹 李磊 刘天喜 陈晖 曲烈 郑晓明 著

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

区块链技术指南 / 邹均等著 . —北京：机械工业出版社，2016.11

ISBN 978-7-111-55356-4

I. 区… II. 邹… III. 电子商务 – 支付方式 – 指南 IV. F713.361.3-62

中国版本图书馆 CIP 数据核字（2016）第 268750 号

区块链技术指南

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码：100037）

责任编辑：高婧雅　　责任校对：殷　虹

印　　刷：中国电影出版社印刷厂　　版　　次：2016 年 11 月第 1 版第 1 次印刷

开　　本：186mm × 240mm　1/16　　印　　张：17.75

书　　号：ISBN 978-7-111-55356-4　　定　　价：69.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

客服热线：（010）88379426　88361066　　投稿热线：（010）88379604

购书热线：（010）68326294　88379649　68995259　　读者信箱：hzit@hzbook.com

本书作者

邹均：中关村区块链产业联盟专家、服务合约（Service Contract）方向博士，关注与实践区块链技术与应用。擅长云计算、大数据、软件定义存储。现为海纳云 CTO，曾任 IBM 澳洲金融行业首席软件架构师、多个云计算公司高管，是融智北京高端外国专家。在国际会议期刊发表论文 20 余篇，获 2015 年澳中校友会 ICT 和媒体类别杰出校友奖，区块链相关论文获 2016 年 IEEE ICWS 最佳博士论文奖。

张海宁：VMware 中国研发中心云原生应用首席架构师，西蒙弗雷泽大学计算机科学硕士，多年软件全栈开发经验，Harbor 企业级容器 Registry 开源项目负责人，Cloud Foundry 中国社区最早的技术布道师之一，国内最早的 iOS 开发者。在 VMware 公司先后负责开源 PaaS 平台 Cloud Foundry、大数据虚拟化、软件定义存储 VSAN 等领域的技术布道和解决方案推广。目前着重关注区块链、容器和云计算等领域的研究和开发工作。之前曾担任 IBM 资深软件工程师、Sun 公司资深解决方案架构师等职务。

唐屹：广州大学教授、理学博士，专注于区块链安全与应用、网络信息安全、分布式计算等，为国外知名安全公司开发过椭圆曲线密码软件，获密码科技进步二等奖（省部级）。主持或参与完成多项国家级或省部级自然科学基金与人才计划等重点项目。

李磊：合肥工业大学副教授，Macquarie 大学博士。擅长数据挖掘、社会计算、智能计算。获 2011 年澳洲最优博士论文提名，并多次担任 IEEE 国际会议的程序委员会委员及组织者。在社会计算和区块链等领域发表论文 40 余篇，被引用 350 余次。

刘天喜：深圳拓邦股份有限公司总经理助理，高级工程师、北京大学博士。在移动通信、集成电路、移动互联网、物联网等领域深耕多年，擅长技术产业研究、行业分析和战略规划，主导或参与中国工程院、中央网信办、工信部、国资委等十余项产业研究课题。发表学术论文 10 余篇。

陈晖：区块链 PPk 开源项目发起人和主要开发者、巴比特网站专栏作者与区块链技术版版主。对网络与通信技术有深入实践与研究，十余年的软件研发和项目管理经验。通过深度实践以比特币为代表的数字加密货币领域，率先提出“区块链 + 网络通信”将最大化发挥区块链革命性价值的观点，并着力以开放开源项目的形式推动区块链与网络通信领域融合的技术创新和应用发展。

曲烈：Macquarie 大学博士，曾任 Macquarie 大学研究员、助教。从事信息安全、密码学、区块链、服务计算以及信息系统等领域的研究。多次在国际知名会议和期刊发表论文，并受邀宣讲。

郑晓明：中国电信云计算分公司工程师、Macquarie 大学博士，专注于云计算、云存储、监控系统、推荐系统、模式识别等，近期研究区块链相关技术。

序一：什么是区块链

2015年是国外区块链的元年，世界许多重大组织，包括高盛、花旗银行、英国央行、美国央行等机构纷纷在区块链上面投资。大量的投资从2015年10月开始便进入了区块链，原因是在《华尔街日报》刊登一篇的文章，里面报道区块链经过了多次的实验和验证，许多金融机构证实了区块链是一个颠覆性的技术。之前华尔街日报甚至宣称，区块链是最近500年以来在金融领域最重要的突破。而这500年来有多少科技上的突破，但华尔街日报却说区块链是人类历史上在金融领域最大的突破。这可能是因为出现了一个新的货币媒介，而每一次新货币媒介出现，都会引发社会和经济上的重大改革。

2016年1月，英国首席科学家建议英国政府把区块链技术列为英国国家战略，这是区块链历史上一个重大突破，原因是基于华尔街以及金融机构对区块链的评价。但自从2016年1月以后，区块链的评价是基于科学历史悠久的英国官方的评价。从各样指标来看，英国在科学上的建树经常是排名第二，仅次于美国。而世界科学排名第二的英国甚至把区块链列为国家战略，表示区块链的重要性毋庸置疑，而且有深远的影响。能够成为国家战略必须在科学上被验证过，另外还必须带来巨大的商业价值，两者都不可缺少才能成为国家战略。笔者曾在2016年3月拜访英国首席科学家，他们认为，区块链可以在各行各业使用，带来行业公平，例如：诚实报税、政府监管、反洗钱、国家安全等。

2016年可以说是中国区块链元年，因为在2016年区块链在中国受到极大的重视。首先是1月的时候，人民银行宣布要使用数字货币。然后在30日以后，许多中国的组织单位就开始投资区块链。中国许多大学也开始研究区块链技术，大型金融机构都纷纷表态成立区块链团队来研究区块链，区块链的讨论班以及研讨会如雨后春笋一般大量涌现。

但到底什么是区块链？笔者在2015年开始研究区块链，就发现了一件事情：学生

们在实验，提出来的区块链模型、算法，或者架构都是有偏差的，而且有时候偏差甚大，例如，在设计私有区块链的时候把公有区块链的全部思想搬过来。结果不像私有区块链，但也不像原来的公有区块链。另外发觉很多人对相关的算法不熟悉，所以有的时候会有一些错误的看法，例如拜占庭将军的问题是一门专门的学问，而区块链只是用了一个近似的算法，若是把两者混为一谈，就会让人感到迷惑。

再加上在讨论区块链时，有时候会有情绪化、宗教化或者政治化的言语出现，原来在数字货币领域，数字货币的先锋常带有一些政治思想，如无政府主义。再加上原来的数字货币过去有洗钱、犯罪的记录，所以在讨论时，有时候会失去焦点。这一点在英国首席科学家的报告里也有提出来，他们认为应该重视区块链，把区块链当做一门科学技术来看，而且是一门有助于经济的科学技术，而不是吹捧任何政治思想，或传递宗教概念。

笔者从今年初开始多次提出应该以系统工程角度来发展区块链技术，例如基于云计算、软件工程、数据库等系统工程技术来开发区块链，区块链不只是一个加密技术或是数字货币，而是一门系统工程。区块链不是某些特殊政治思想的乌托邦，或洗钱的工具，而是一门科学家和工程师可以研究的系统工程，而且这项技术可以成为国家战略，改变各行各业的流程以及基础设施。英国首席科学家已经做出这样的判断，英国央行也做出了类似的决定，英国政府已经派了两位部长来领导这项计划，这就是我们所期待的。

所以我非常高兴像邹均、张海宁、唐屹、李磊、刘天喜、陈晖、曲烈、郑晓明这些年轻的学者们开始书写区块链技术，因为现在市面上有关区块链的书都是在讲解区块链的概念及应用场景，但是今天描述区块链技术的书却很少。我们希望读者能多了解区块链技术，多发展区块链技术，并且加以应用。只有我们了解区块链技术之后，才能真正理解区块链的意义，而不会随波逐流，人云亦云，并且有自己的判断，希望读者们能够认真读这本书，了解区块链技术，相信必定会大有收获。

蔡维德

美国亚利桑那州立大学荣誉教授，北航区块链实验室主任

序二：区块链——未来已来，只是尚未流行

比特币诞生于2008年美国次贷危机的末期。在比特币白皮书，即中本聪的论文《比特币：一种点对点的电子现金系统》中，还没有“区块链”这个词，只有“区块”（Block）和“链”（Chain）。一些人为这种超越主权、不会滥发的虚拟数字货币而欢欣鼓舞，开始积极投入到挖矿、炒币中，甚至发行自己的数字货币进行筹资（ICO），俗称“币圈”。而另一些人，包括很多专家和学者，则专注于比特币底层技术，对区块链（Blockchain）技术和应用进行深入地研究，考虑能否将这个技术加以改进，运用到更多的领域中去，俗称“链圈”。

七年之后，以2015年10月美国《经济学人》杂志发表的《信任的机器》（The Trust Machine）的封面文章为标志，大家意识到，作为比特币底层技术的“链”，其价值远大于比特币本身。区块链可以让人们在没有中央权威机构监督的情况下，对彼此的互相协作建立起信心。简单来说，它是一台创造信任的机器。华尔街开始热捧区块链。Gartner发布的2016年技术炒作曲线图表明，当前区块链正处于期望的最高点，即“过度期望期”，这也意味着在未来不久的一段时间，区块链将坠入“期望幻灭期”。人们对区块链的过度期望，实际暗示着对其存在很多误解，其中最典型的有三个，因为其关键词的首字母都是D，所以笔者将其归纳为“3D误区”。

误区一——区块链是一种颠覆性（Disruptive）的新技术

首先，区块链不是一项新技术，而是一个新的技术组合。其关键技术，包括P2P动态组网、基于密码学的共享账本、共识机制（拜占庭将军问题，即一种分布式场景下的一致性问题）、智能合约等技术，都是已经有十年以上的老技术了。但是，中本聪将这些

技术很巧妙地组合在一起，并在此基础上引入了完善的激励机制，用经济学原理来解决传统技术无法解决的问题。

其次，这个技术组合虽然有其独到的创新之处，但并非是颠覆性技术，是现有技术的有力补充。目前大部分人已经认同，区块链是“价值互联网”的基础协议，从这个角度看，其地位与当前“信息互联网”的 HTTP 协议相当，两者都是建立在 TCP/IP 协议之上的应用层协议，同是互联网的两大基础协议。因而，两者是互补而非颠覆的关系。

最后，这个技术组合，并未颠覆现有业务，而是引入了新的思想，去改善和改造现有业务模式，从而为大众提供更好的、普惠的服务。《华尔街日报》在 2015 年 1 月曾发表题为《比特币与数字货币的颠覆性革命》的文章，认为比特币的数字货币发行机制可能“颠覆”目前各国央行的法定货币发行模式，这算是最接近“颠覆”性的区块链案例。而实际上，比特币在经过 8 年多的发展后，虽然总市值发展到了 100 亿美元，但在全球经济活动中的比重还是微不足道。与此同时，也确实有一些国家的央行，如英国和中国，在考虑摈弃比特币的挖矿机制后，通过借鉴数字货币的一些机制，在一定范围内实现可跟踪、可追溯、数字化的法定货币。

误区二——区块链就是去中心化（Decentralized）的

首先，很多人认为 Decentralized 是区块链的核心特征，并将其翻译为“去中心化”。然而这个最早由国内“币圈”所做出的翻译，多少有一点主观和政治化的色彩。作为软件系统的网络架构一般有三种模式：单中心、多中心、分布式。单词 Decentralized 只是表明不是单中心模式，可能为多中心或弱中心，也可能是分布式的。所以在中国台湾地区，大多将 Decentralized 翻译为“分散式的”而不是“去中心化的”。

其次，在中本聪的整篇论文中并没有提到过 Decentralized，而只有 Peer-to-Peer（P2P）。在 2016 年 6 月召开的 W3C 区块链标准会议上，以太坊的核心开发团队 EthCore 就明确表示，不再使用 Decentralized 这个词，而是用 P2P、Secure、Serverless 这类纯技术性词语。

最后，The DAO 事件表明，完全去中心化是不可行的。The DAO 是一个基于以太坊公有链的众筹项目，它在短时间内就募集了价值 1.6 亿美元的数字货币，成为史上最大的众筹项目。然而由于其智能合约的漏洞，导致 The DAO 被黑客攻击并转移走价值 6000 万美元的数字货币，最后不得不黯然落幕。在挽回这个损失的过程中，原有的去中

心化机制未能解决问题，最后还是通过“集中式”的方式，强制以太坊进行“硬分叉”完成交易回滚。但这也导致了以太坊社区的分裂，产生了 ETH 和 ETC 这两种同源却又不同价格的数字货币，给以太坊生态系统带来了很多负面影响。此次事件之后，很多人对区块链的“去中心化”进行了反思。前上交所总工、ChinaLedger 联盟技术委员会主任白硕则认为“去中心化不是区块链的本质特征”。万向控股副董事长兼执行董事肖风则进一步阐述“区块链的核心是分布式而不是去中心”。

误区三——区块链交易存在很大的延迟（Delay）

在使用比特币进行支付时，一般需要 10 分钟才能完成一次支付确认。如果要保证支付交易的不可逆转，通常需要等待连续的 6 个数据块完全确认，这至少需要 1 个小时的确认时间。而我们通常使用的银行网银支付和第三方支付，通常都是秒级完成的。与之相比，使用区块链的比特币支付实在太慢。

然而，我们再考虑一下跨境支付的场景，当我们使用 Swift 完成一次跨境汇款时，通常需要 3 ~ 5 个工作日，对方才能收到相应的款项。而使用比特币进行跨境汇款，仅仅需要一个小时就能收到汇款。如此比较起来，比特币支付已经是非常快了。

为什么有两个完全不同的结论？因为，对于比特币支付来说，支付确认过程即是清算和结算的过程。如果把支付过程和清结算过程作为一个整体，来比较两类支付的延迟时间，使用区块链进行交易还是很快的。区块链交易的本质，是大幅减少了交易后的处理工作，消除了大量的人工干预过程，从而提高了交易效率。

通常我们把区块链分为公有链、私有链、联盟链三种，比特币和以太坊都属于公有链范畴。在数字货币之外的场景中，尤其是在金融领域中引入区块链技术，将面临很多问题。如何引入以及引入哪种区块链，还存在许多权衡决策方面的障碍。

第一，主流金融机构难以接纳公有链。R3 发布最新研究报告，证明公有区块链不可作为金融机构解决方案。2016 年 Swift 发布白皮书指出，当前世界主流金融机构无法接纳公有区块链。对于这些金融机构而言，需要的是一个自主可控的系统，而公有链显然做不到这点。

第二，私有链与公有链架构差异大。笔者曾仔细分析了以太坊和超级账本这两个典型区块链的模块结构，发现两者差异巨大。很多公有链的核心模块，如挖矿、PoW 共识、原生货币等，在私有链环境中是完全不必要的，甚至是有害的。与此同时，公有链系统

中还缺失一些诸如身份认证、权限管理等在私有链中必要的模块。以太坊创始人 Vitalik 也曾坦言，只有 5% 的以太坊程序可被金融领域使用。[⊖]

第三，私有链和联盟链还很不成熟。目前，以比特币和以太坊为代表的公有链相对比较成熟，而私有链和联盟链则远远不够成熟。开源而且好用的联盟链，更是不存在。目前全球影响力最大的开源联盟链，是 Linux 基金会下面的超级账本（Hyperledger）项目，目前已有 95 个成员单位。旗下的 Fabric 子项目是以 IBM 捐献出的 OpenBlockchain 为主体搭建而成的，目前还处在 0.6 版的快速迭代过程中，到 0.8 将是 Alpha 版，而 0.9 则是 Beta 版，再经过 3 个 RC 版本之后，才会进入相对成熟的 1.0 版。

想要找到或研发出一个成熟稳定的、适合金融领域的联盟链底层系统，还任重道远，需要很多仁人志士的共同努力，踏踏实实地投入到区块链的基础研究中去。

在目前已出版的区块链书籍中，有很多都冠以“革命”、“重塑”、“重新定义世界”等煽动性词语作为书名，这更像是一种口号，而非切合实际的研究。我很高兴地看到，还有像邹均、张海宁、唐屹、李磊、刘天喜、陈晖、曲烈、郑晓明等这些研究者们，在踏踏实实地研究区块链底层技术，用普实的话语来介绍和普及区块链技术，让更多的人了解和接受区块链技术，实实在在地让人们了解区块链技术特征和特点，以及在现阶段环境下的不足，如何去改善这些不足等。知己知彼，方能百战不殆。世上没有“银弹”，没有哪一种技术能解决所有的问题。

希望读者们能够通过本书，深入地了解区块链技术。也只有深入了解其底层运作机制和原理，才能更好地灵活运用该技术，取得理想的效果。

未来已来，只是尚未流行，我辈仍需多努力。

张斌，联动优势科技有限公司 CEO

⊖ 参见《金融电子化（2016.5）》P60，《区块链技术在金融领域的应用解析》。

序三：区块链——连接虚拟与现实

我们对于一种新兴的技术，往往会在短期内对它有过高的不切实际的期望；泡沫破灭后，在长期的时间轴线上，又往往会忽视它的深刻影响，这一句话，用在区块链上，再合适不过。

区块链的发明，是建立在互联网之上。其所使用的技术，像 P2P、分布式存储、分布式密钥的思想，十几年前就已经存在，但是如果没有中本聪那一篇开创性的关于比特币的白皮书，所有这些强大的工具，都还只是埋藏在学术论文堆里。因为这些工具单独使用，并不能解决问题，只有中本聪，出人意料地提出了一个系统性的、可供实践的解决方案。如果他能提前十年提出这篇论文，那么比特币就可以提前十年发明出来。所以，单个技术点，并非是区块链的魅力所在，运用这些技术的全新思想，才是区块链的本质和核心。

单纯把区块链等同于一种分布式数据存储技术，就像将浏览器说成是一个网页解释器，将手机说成是一台手持电话，将云计算说成是一个服务器的集群一样，说了等于没有说，甚至比没说更糟糕，更容易造成误解。当全球的用户都打开浏览器访问网页，当街上每一个人都携带着一台能拍照、能上网、带 GPS，运算性能可以发射登月火箭的智能手机，当我们所有的工作和生活数据都发生与存储在云上的时候，我们看到在浏览器、移动互联网和云计算上所承载的产业生态，跟最初的技术描述相比不知道差了多少万里。所以有人让我用一句话解释什么是区块链的时候，我往往会争取机会多说几句，争取让人更多了解一点。

从功能上说，互联网实现了信息的传播，而区块链实现了价值的转移。互联网在最开始的时候，就是以信息传输管道的模式进行的设计，TCP/IP 协议底层并不关心上面传输的数据有什么差别——对于底层的交换机和路由器来说，一切都是 0 和 1 而已。无差别的信息传输，创造了信息复制的便捷通道，也造就了今天信息爆炸的信息社会。但是互联网虽然解

决了信息传播的问题，却带来了信息权属的新问题，我们可以将一首歌曲或者电影，在几个小时内传遍全球，我们却不能知道，究竟是谁拥有这部电影的权利，是通过什么样的路径进行的传播。而区块链则可以做到，我将一个数据，发送给另外一个人之后，我自己就不再拥有这个数据的所有权，从而实现了可以利用一个虚拟的系统，来传输实际的价值。

从机制上说，如果说 TCP/IP 是机器与机器之间的通信协议，而区块链就是机器与机器之间的信任机制和合作协议。对于不需要验证真假的信息传输来说，TCP/IP 已经足够可用，但是一旦属于不同实体的计算机，需要彼此之间进行自动化的沟通和合作的时候，问题就会变得相当复杂。现实世界公司与公司之间的合作，有律师和合同来进行条款约定，有执法机关来保障合同的实行，而在虚拟世界，计算机没有办法开设银行账户，属于不同实体的计算机，也没有办法去法院起诉对方，因此在沟通和合作的时候，一定要有一种有效的机制，来快速实现共同协作。区块链就可以起到这样一个作用，所以在区块链行业中有一句话：代码即法律（Code is the Law）。未来不管我们的生活还是工作，都会有越来越多地需要计算机参与，人类将整体进入后人工智能时代，区块链就是在为这个时代的到来进行前期的铺垫和准备。未来我们将会看到无人驾驶汽车，通过区块链协议自动缴纳过路费用；智能投资顾问自动为我们计算各种投资组合；未来最先进的金融公司，也会像现在的无人工厂一样，看不到太多工作人员，只有无数的计算机，在快速地缔结无数的智能合约，进行精确到小数点后的资产配置。

因为区块链的以上属性，区块链将会是连接虚拟世界与现实世界的最佳桥梁。在未来，区块链所连接的，不会像比特币一样是无法辨别的匿名账户和价值不定的虚拟资产，而将会是千千万万真实存在的个体和公司实体。上面所承载的资产，都将具有现实的价值和对应物，而这个虚拟的网络上发生的一切，也都会直接作用于现实世界。这一过程，需要的不仅仅是单纯的技术，还需要金融、商贸、法律、政府等各方面专家和人才凝聚在一起，来保证这一映射的有效性，也是我们一直在努力推进区块链生态系统和可信区块链概念的原因。区块链有巨大的潜力和未来，而这些潜力和未来，需要社会的共识与力量来共同推进和实现。

邓迪

太一云科技有限公司董事长兼 CEO

序四：区块链——转型之擎

邹均先生在国内外企业的IT架构、云计算、大数据、IT产品创新方面有很多年的经验，邹均本人也是我多年的好朋友和同事。这次邹均先生主写的这本区块链的书，相信一定会在IT业内，特别是在企业IT架构圈内产生巨大的反响，一定会深受广大区块链爱好者、参与者、实践者的热烈欢迎。

我和邹均先生工作背景相似，曾经从事过多年企业IT工作，从2009年开始，做云计算的创新，近年来也做金融科技的创新。从我这一年多时间的区块链的实践中，我个人看到区块链目前虽然还在发展初期，而每天区块链技术都有新的变化和突破，每天都是“山雨欲来风满楼”。但是区块链这样一个意义重大的技术，对整个IT的架构、基础协议、标准、运营、环境具有颠覆性的意义。因此我们应当充满紧迫感，应当预先了解区块链技术、商业模式和发展趋势，加强与国内外各界的合作，特别是在区块链的底层领域、区块链的平台领域和区块链的应用领域的合作，我们应当在区块链的全球协议和标准方面要占据主动。

区块链技术具有全新的理念和逻辑结构，并且它每天还处在发展变化过程中，因此区块链技术与应用在企业内不可能单打独斗，区块链的应用必须在企业架构中上着天、下着地，和企业现有的应用系统相互关联。我们不应该简单地把区块链理解为一项技术，而应当考虑它在更高的企业IT架构转型层面的作用。区块链的应用不是简单地提供一个只能追加、不能更改的分布式数据库解决方案，而是要把区块链与云计算、大数据和传统企业的系统相互关联，使得企业系统由原来的传统系统和云计算这种“双核驱动”转变为传统系统、云计算与区块链的“三核驱动”，让企业的异构系统更好地发挥协同效应，一起解决原来传统IT系统难以解决的问题，这样才能更好地发挥区块链的

独特性，才能够使传统企业 IT 架构更好地转型。

本质上，因为区块链链与链之间具有隐私、安全、共识、自治、价值共享的特性，所以在技术层面解决了互联网上的价值传递问题。同时，区块链又具有底层开源和改变业务规则、创新业务多方共识等逻辑，因此区块链是未来整个 IT 架构和互联网转型的重要支撑。而企业与互联网 IT 架构的转型也为未来经济的转型、服务模式、信用交换和商业规则的转型提供了关键支持，因此研究和应用区块链不仅要研究技术，更要注意在互联网时代赢者通吃的规则，重要的是要研究和应用区块链带来的商业规则的改变。

以前我们的信息化，不管是企业信息化、政府信息化，还是个人信息化，实际上都侧重在机构内部的信息化。这几年随着互联网、云计算、大数据、平台经济的蓬勃兴起，现在 IT 正在促使企业由内部信息化转型为外部信息化，最终通过平台转型为信息化的企业，由政府信息化转型为信息化政府，由个人信息化转型为信息化个人，这些词虽然相似，但性质具有很大的不同。它们在逻辑关系、业务处理方式、信息的确权、信息的使用、组织流程的改变、企业治理结构方面有很大不同，信息化已经不再是工具、手段和渠道。这样一个信息化平台的升级，未来会使得实体经济更好虚拟化，使得虚拟经济更好地结合实体化。

实施区块链既需要具有传统 IT 系统的经验，也需要有互联网、云计算、大数据的实施经验，需要对整个 IT 系统变迁具有很强的洞察力，需要把整个 IT 系统协同起来，让整个 IT 系统互联互助，相互合作。因此，区块链系统在企业的应用，必然需要结合本地的实践，发挥原创的精神，必然还要有互联网时代产品开发的能力。而做一个好的区块链应用更需要研究共享经济理论、价值互联网和金融科技的创新与发展。这一切都需要在区块链理论与研究方面走到前列。

因此，我希望邹均先生等人写的这本区块链的书籍，会连接 IT 架构的过去、现在与未来，开启大家创新的热情，会对行业产生影响，同时为大家开启一扇协同企业传统系统、云计算、大数据和区块链新的大门。

黎江

北京世纪互联创新研究院院长

前言

为什么要写这本书

1900 年 9 月 8 日，一场 4 级强度的飓风横扫德克萨斯州的加尔维斯顿。这个位于墨西哥湾的岛城，靠近德克萨斯海岸，在灾难来临前拥有 37 000 人口和光明的经济前景。飓风猛烈攻击了这个毫无防备的低海拔城市，给该市带来了巨大的毁坏。飓风风速为每小时 225 千米，毁掉了 3600 座建筑，使占整个城市 3/4 的 12 个街区彻底消失，死亡人数为 8000 ~ 10 000 人。是迄今为止，美国历史上死亡人数最多的自然灾害。

而 2016 年 8 月 2 日在中国华南沿海登录的“妮妲”台风，风力 14 级，最高风速每小时 151.2 千米，台风过境的广东、广西、湖南、贵州、云南 5 省（自治区），虽然也造成了重大经济损失，但在人员伤亡统计报告中，只有 1 人失踪。

这两次自然灾害的结果如此不同，归功于人类掌握了计算这个神奇工具。在妮妲形成过程中，美国、日本、中国气象监控部门就不断跟踪，通过监控数据，气象数学模型和强大的计算能力，对台风进行了准确的预报和预警。在台风到来前，有关部门做了积极准备，7.6 万人得以紧急转移安置，使得损失得以降到最低。

今天，IT 已经渗透到各行各业，人们已经能近距离接触无人驾驶、机器人、虚拟现实（Virtual Reality）、增强现实（Augmented Reality）等先进技术，当人们在享受 IT 给人们生活带来的各种便利和好处的时候，也日益感受到来自不当使用科技所带来的挑战。例如，国内日益猖獗的电信诈骗，全球范围内黑客的攻击和安全勒索，以及未来基因技术和 AI（人工智能）技术给人类所带来的伦理、生活和工作方面的全方位冲击，都使得有识之士开始思考如何应对科技发展所带来的风险。

一直以来，笔者对计算技术有一种既感恩又敬畏的情结。首先感恩我们的时代，计

算技术的发展使我们避过很多前人无法避过的灾难；但高速发展的计算技术必然导致机器的智能超过人类自身，因此而产生的未来不确定性也使笔者的敬畏之心油然而生。

笔者也一直有一个预感，未来可能需要针对 IT，特别是与业务结合紧密的云计算和智能设备建立监管、问责的机制。笔者的意思不完全是对从事 IT 或智能设备的人进行监管问责，甚至要考虑对智能设备进行自动问责。这个看似荒谬的想法促使笔者选择了云计算的问责机制（Accountability in Cloud Services）作为博士研究方向。

所谓云计算的问责机制（Accountability），指的是在云计算架构中，能建立一个自动化的问责机制。该机制包括形式化的标准服务合同定义，服务合同的发布，服务合同执行的监控，合同违约方的自动发现，违约方的罚则和执行，以及合同双方争议的仲裁。举个例子来说，今天公有云的提供商，都没有提供能让电脑理解的云服务合同。合同双方的责任、义务和权利没有精确的界定；云服务提供商的服务好坏，是否遵从合同，都没有自动化的方法去检测；服务故障责任也没有办法界定；出现争议也只能靠人工去解决。而云计算的问责机制，旨在建立一个自动化的体系来让电脑自动规范电脑的行为。

可想而知，这个研究课题非常有挑战。在博士研究的过程中，笔者也走了很多弯路，一直没有找到好的解决方法，直到三年前接触到比特币，突然意识到区块链技术是提供问责机制的最理想平台。这是因为区块链技术中的防伪、防篡改、交易可追溯、数字签名和智能合约技术提供了一个公正、可问责（Accountable）、自动执行的技术平台基础。

但是区块链目前还停留在概念炒作阶段，很多关注点还停留在金融应用，特别是虚拟货币方面的应用。笔者认为，区块链未来可能最适合作智能设备的“警察”，为物联网和智能设备的自治管理提供一个基础平台。区块链技术应该推广应用到除金融外的行业，因此萌生了写这本书的念头，作为博士研究工作的一个延续。

而写这本书的另一个原因，也是深感在学习区块链技术过程中碰到的参考资料不足的痛苦，希望能整理过去的学习所得，对区块链初学者有所帮助。

从 2008 年中本聪发表比特币白皮书算起，区块链技术才走过短短 8 年的时间。虽然区块链 1.0、2.0 和 3.0 的架构理念已经提出并得到一定程度上的认可，但区块链的技术发展仍然处于初级阶段，区块链的应用还刚起步，成熟的区块链应用除了比特币系统，还寥寥无几。在这种情况下写关于区块链的书籍，其实面临一个两难境况。一是区

块链的技术变化快，像个移动的靶子；可供参考的资料又少，要准确把握一个快速变化的技术非常困难，而且受限于写笔者的水平，实践经验，写出来的书难免有很多错误，弄不好会贻笑大方。而另一方面，正因为变化快，资料少，广大区块链技术爱好者又渴望能找到一本对他们学习、理解、掌握区块链架构和技术有所帮助的书。

目前在市场上的区块链书籍大致分为两类：一类是以梅兰妮·斯万（Melanie Swan）的《区块链：新经济蓝图及导读》为代表的，谈区块链对整个宏观层面所带来的革命性影响的战略性书籍；一类是以安德鲁·安东普洛斯（Andreas M. Antonpulos）的《精通比特币》，以及普林斯顿大学以阿文·拿瑞延南（Arvind Narayanan）为首编著的《比特币和密码学技术》为代表的专注于比特币的技术性书籍。这些书籍满足了目前市场上一部分对区块链在行业中的应用有兴趣的偏业务的人士，以及对比特币技术有兴趣的偏技术的人士的需求。

在这两类书籍所覆盖的市场中，其实还有一个很大的空白。我们发现，在对整个区块链架构（包括区块链 1.0、2.0 和 3.0）进行系统性剖析，包括对其中关键技术（密码学、共识算法）等进行系统性论述，对不同的区块链架构形式（联盟链、公共链、私有链、侧链、多链、互联链等）进行系统性介绍的书好像还没有。而这样的书对理解、普及区块链技术，推动区块链应用落地可能会有所帮助。因此，与其等待这样的书籍出现，不如自己行动，为区块链技术的推广尽绵薄之力。笔者也就自不量力，把可能被同行笑话的风险置之脑后，鼓起勇气集合几个对区块链着迷、志同道合的朋友，在条件不成熟，时间比较仓促的情况下，经过不少不眠之夜的努力，克服重重困难，特别是在机械工业出版社华章分社编辑高婧雅的大力协助下，完成了该书。

本书的缺点是显而易见的。

一是因资料匮乏、技术变化快而难免出现技术错误。因此，本书的目的，主要是抛砖引玉，欢迎读者多提宝贵意见，争取在下一版本能纠正大部分的错误，不断完善、提升本书的质量。

二是缺少应用案例。其实目前网上的应用案例也有不少，但是我们认为，如果只是拿别人在网上的案例加工修改，从深度、广度方面都经不起推敲，起不了真正案例的作用。除非由真正落地该应用案例的主要负责人来写，才能使读者有真正的收获。受限于我们的人脉圈子和条件，目前只能请到 PPKpub.org 开源社区组织者陈晖先生来写一

个区块链在标识注册方面的应用案例。在此鸣谢陈晖先生的大力支持，将来也欢迎有更多的区块链应用的领军团队提供应用案例，在未来更新的版本中补上在应用案例方面的短板。

本书特色

1）和目前市场上主流的区块链书籍强调区块链去中心化的概念，以及对业界带来的革命性影响不同，本书主要是从技术的角度，介绍区块链的基础概念，特别是对区块链的架构进行了详细的剖析。

2）对区块链的关键技术，包括区块链架构（1.0、2.0、3.0）、密码学和共识算法等做了一个详尽的介绍。

3）提供了比特币开发指南，通过以太坊智能合约开发来帮助初学者入门。本书也用专门一章来讨论区块链的常见问题，包括对近期发生的 DAO 攻击事件，都有详细的分析。

4）在区块链技术落地方面，本书也提供比较典型的区块链解决方案，包括支付和标识登记方面的解决方案。

5）以独特的架构演进对 IT 发展的影响为切入点，给读者展示一个全新观察整个 IT 历史的视角，并在这个视角下探讨区块链技术在未来 IT 发展中的影响和地位。

本书中一些实操的例子和章节，比较适合区块链初学者和程序员，可以成为区块链入门的书；架构剖析和深入分析方面的章节，比较适合 IT 架构师，以及区块链技术爱好者来深入了解区块链架构特点和技术细节，对设计区块链的解决方案有所帮助；解决方案和常见问题章节有助于区块链从业人员全面了解区块链应用落地方面的情况。最后一章是从架构视角对 IT 发展的一些观察，仅供喜爱思考的 IT 从业者参考。

读者对象

- ❑ 区块链从业者
- ❑ IT 架构师
- ❑ 区块链应用开发人员
- ❑ 对区块链技术感兴趣的人员

如何阅读本书

本书分为三大部分，共 11 章。

第一部分介绍基础和入门，包括以下 2 章内容。

第 1 章 本书的开篇，首先介绍区块链的定义和特点，并简单介绍了区块链的主要类型，然后通过介绍购买、存储和交易比特币等实际使用场景来让读者对区块链有所体验，然后再探讨一些关于区块链的常见问题。

第 2 章 介绍区块链的基础概念，为后面深入介绍区块链技术做铺垫。

第二部分介绍架构和核心技术，包括以下 8 章内容：

第 3 章 详细介绍区块链 1.0、2.0、3.0 典型架构，同时介绍了互联链的概念和架构。

第 4 章 详细介绍了区块链涉及的密码学原理和典型的算法。

第 5 章 介绍了在区块链架构中常用的共识算法。

第 6 章 提供比特币开发指南，通过实际案例来帮助初学者入门。

第 7 章 提供以太坊上的智能合约开发指南，帮助初学者掌握智能合约的开发要领。

第 8 章 详细介绍 HyperLedger 开源项目及其架构。

第 9 章 讨论区块链上常见的问题，包括最近出现的 The DAO 攻击的源码级分析。

第 10 章 讨论区块链上的典型解决方案，一个是以闪电网络为主的支付方案，另一个是以标识登记为主的开源 ODIN 解决方案。

第三部分为回顾和展望，即第 11 章，主要回顾 IT 架构演进历史并展望未来区块链对 IT 发展的影响。

勘误和支持

由于笔者的水平有限，编写时间仓促，书中难免会出现一些错误或者不准确的地方，恳请读者批评指正。如果你有更多的宝贵意见，欢迎通过微信或邮件进行讨论。你可以通过微信 joezou3986、微博 @ 云中君 3986，或者发送邮件到邮箱 joezou@openstack.org.cn 联系到我，期待能够得到你们的真挚反馈，在技术之路上互勉共进。

致谢

首先感谢我的作者伙伴——张海宁先生、唐屹教授、李磊教授、刘天喜博士、陈晖

先生、曲烈博士和郑晓明博士。他们在工作之余，挤出宝贵时间为本书贡献了他们对区块链技术的理解和洞察。特别感谢我的大学同门师弟 Henry 张海宁先生在关键时刻的出手相助，为本书贡献了很多精力，他不单在内容上积极供稿，也在本书的审定、修改和校正方面下了很多工夫。唐屹教授和李磊教授也在繁忙的教学和学术研究中抽出时间来对一些区块链的基本概念和关键技术（包括密码学和共识算法）做了详尽的阐述。刘天喜博士在本书的框架规划和开篇设计上做了很大贡献。而陈晖先生的比特币开发指南对很多初学者入门有很大的帮助，他的 ODIN 开源项目也是区块链登记方面的一个典型解决方案。曲烈博士的智能合约开发章节给众多以太坊开发初学者提供一个易懂、易上手的应用指引。郑晓明博士也对主流代币做了比较全面的介绍。

本书作者也得到中关村区块链联盟的大力支持，在此也特别鸣谢中关村区块链产业联盟秘书长王安平先生、副秘书长范金刚先生和林大鹏先生以及联盟发展部张培部长。同时也感谢江源老师、江苑绛博士，他们的鼓励成为我坚持下来的动力。另外在写书过程中也得到澳洲富士通区块链技术架构师董仲利先生、信达证券区块链首席专家曹寅先生、亚投行企业 IT 项目管理专家 Allen 邵以及合肥工业大学刘古刘和方辉先生的帮助，在此对他们表示感谢。

另外感谢比特币开源社区、以太坊开源社区，以及巴比特社区的各位技术专家们的博客文章，每次阅读必有所获，本书也多处引用了他们的观点和思想。

非常感谢机械工业出版社华章公司的编辑高婧雅，她的敬业精神和编辑效率令我由衷敬佩，她的反馈、建议、鼓励和帮助引导我们克服诸多困难完成全部书稿。

特别致谢

最后，因为工作和写书，牺牲了很多本该陪伴家人的时间。我要特别感谢我的母亲从小对我的培养，也要感谢我的哥哥姐姐们在儿时营造的和睦互助、求知好学的家庭环境，这对我长大以后形成对新兴技术浓厚的求知欲性格有很大影响，一直以来在我的职业生涯中都受益匪浅。更要感谢我太太 Annie 长期以来对我的默默支持，以及女儿 Beverley，儿子 Skyler 对我工作的理解。

谨以此书献给我最亲爱的家人，多年以来帮助、支持我的师友们，以及众多热爱区块链技术的朋友们！

我想和作者聊聊

如果你想和本书作者沟通，可以通过以下方式。

1）微信群“区块链技术交流群”，添加群助理微信号 xiaodanmyd 入群。

2）QQ 群“区块链技术交流群”，群号 375936045。

3）关注微信公众号“链信 Chain2Trust”。

4）邹均微信号：JoeZou3986，添加请注明沟通事项。

邹均

目录

第1章

区块链和比特币初体验

区块链（Blockchain）是近年来最具革命性的新兴技术之一。区块链技术发源于比特币（Bitcoin），其以去中心化方式建立信任等突出特点，对金融等诸多行业来说极具颠覆性，具有非常广阔的应用前景，受到各国政府、金融机构、科技企业、爱好者和媒体的高度关注。

在本章中，我们首先介绍区块链的定义和特点，然后通过介绍购买、存储和交易比特币等实际使用场景来体验区块链，最后再探讨一些关于区块链的常见问题。

1.1 区块链简介

2016 年 1 月 20 日，中国人民银行官方网站上发表了一条题为《中国人民银行数字货币研讨会在京召开》的新闻[1]，这一消息迅速在各大主流新闻媒体和比特币、区块链爱好者社区中传播，成为推动区块链技术在国内迅速升温的“导火线”。这是自从 2013 年 12 月 5 日中国人民银行、工信部、银监会、证监会和保监会五部委联合发布《关于防范比特币风险的通知》[2] 以来，相关首次公开对比特币底层技术——区块链技术给予了高度评价。

在我们开始区块链体验之旅之前，让我们简要介绍区块链的定义和其发展历程。

1.1.1 区块链起源——比特币

区块链的英文是 Blockchain，字面意思就是（交易数据）块（Block）的链（Chain）。

区块链技术首先被应用于比特币，如图 1-1 所示。比特币本身就是第一个，也是规模最大、应用范围最广的区块链。

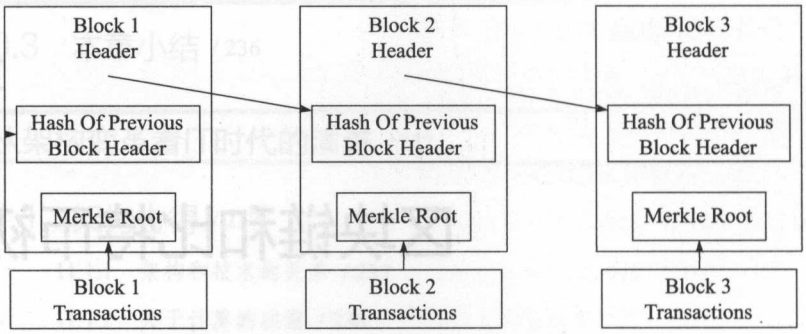


图 1-1 简化的比特币区块链示意图

1.1.2 区块链和区块链技术的涵义

目前，关于区块链没有统一的定义，综合来看，区块链就是基于区块链技术形成的公共数据库（或称公共账本）。其中区块链技术是指多个参与方之间基于现代密码学、分布式一致性协议、点对点网络通信技术和智能合约编程语言等形成的数据交换、处理和存储的技术组合。同时，区块链技术本身仍在不断发展和演化中。

1.1.3 区块链分类

以参与方分类，区块链可以分为：公开链（Public Blockchain）、联盟链（Consortium Blockchain）和私有链（Private Blockchain）。从链与链的关系来分，可以分为主链和侧链。而且，不同区块链还可以形成网络，网络中链与链的互联互通，产生互联链（Interchain）的概念。

1. 公共链

公共链对外公开，用户不用注册就能匿名参与，无需授权即可访问网络和区块链。节点可选择自由出入网络。公共链上的区块可以被任何人查看，任何人也可以在公共链上发送交易，还可以随时参与网络上形成共识的过程，即决定哪个区块可以加入区块链并记录当前的网络状态。公共链是真正意义上的完全去中心化的区块链，它通过密码学保证交易不可篡改，同时也利用密码学验证以及经济上的激励，在互为陌生的网络环境中建立共识，从而形成去中心化的信用机制。在公共链中的共识机制一般是工作量证明（PoW）或权益证明（PoS），用户对共识形成的影响力直接取决于他们在网络中拥有资源

的占比。

公共链通常也称为非许可链（Permissionless Blockchain）。如比特币和以太坊等都是公共链。公共链一般适合于虚拟货币、面向大众的电子商务、互联网金融等 B2C、C2C 或 C2B 等应用场景。

2. 联盟链

联盟链（Consortium Blockchain）仅限于联盟成员参与，区块链上的读写权限、参与记账权限按联盟规则来制定。由 40 多家银行参与的区块链联盟 R3[3] 和 Linux 基金会支持的超级账本（Hyperleder）[4] 项目都属于联盟链架构。联盟链是一种需要注册许可的区块链，这种区块链也称为许可链（Permissioned Blockchain）。

联盟链的共识过程由预先选好的节点控制。一般来说，它适合于机构间的交易、结算或清算等 B2B 场景。例如在银行间进行支付、结算、清算的系统就可以采用联盟链的形式，将各家银行的网关节点作为记账节点，当网络上有超过 2/3 的节点确认一个区块，该区块记录的交易将得到全网确认。联盟链可以根据应用场景来决定对公众的开放程度。由于参与共识的节点比较少，联盟链一般不采用工作量证明的挖矿机制，而是多采用权益证明或 PBFT（Practical Byzantine Fault Tolerant）、RAFT 等共识算法。联盟链对交易的确认时间、每秒交易数都与公共链有较大的区别，对安全和性能的要求也比公共链高。

联盟链网络由成员机构共同维护，网络接入一般通过成员机构的网关节点接入。联盟链平台应提供成员管理、认证、授权、监控、审计等安全管理功能。

2015 年成立的 R3 联盟，旨在建立银行同业的一个联盟链，目前已经吸引了 40 多个成员，包括世界著名的银行（如摩根大通、高盛、瑞信、伯克莱、汇丰银行等），IT 巨头（如 IBM、微软）。

银行间结算是非常碎片化的流程，每个银行各自有一套账本，对账困难，有些交易有时要花几天才能校验和确认。同时，其流动性风险很高，在监管报送方面非常繁琐，也容易出现人为错误，结算成本很高。

针对这种情况，R3 联盟构建了一个银行同业的联盟链以解决这些问题。利用区块链技术，银行同业间可以共享一个统一的账本，省掉对账的繁琐工作，交易可以做到接近实时的校验和确认、自动结算，同时监管者可以利用密码学的安全保证来审计不可篡改的日志记录。

R3 联盟将开发 Corda 分布式账本来实现未来愿景。Corda 的名字来源有两个，该名字前半部分听起来像 accord（协议），后半部分来自于 chord（弦，即圆上两点间最短的

直线）的定义。这个圆就代表R3联盟中的银行机构。从目前公开的资料来看，Corda具有以下特点：

- ❑ 数据不一定要全局共享，只有满足合法需求的一方才能在一个协议里访问数据；
- ❑ Corda不用一个中心化的控制就可以编排联盟成员的工作流；
- ❑ Corda对联盟成员之间的每笔交易形成共识，而不是在联盟机构的系统层面形成共识；
- ❑ Corda的设计直接支持监管者监督和合规性监控；
- ❑ 交易由参与交易的机构进行验证，而不会报告与交易无关的机构；
- ❑ 支持不同的共识机制；
- ❑ 明确记录智能合约与用书面语言撰写的法律文件之间的关联；
- ❑ 采用工业标准的工具来构建Corda平台；
- ❑ 不设虚拟货币。

Corda平台注重互操作性和渐进部署，不会将保密信息发布给第三方。一个机构可以和对手机构看到一组协议，并可以保证对手机构看到的是同样内容，同时报送给监管机构。Corda包括共识、校验、独一性、永恒性和认证等功能。

3. 私有链

私有链则仅在私有组织使用，区块链上的读写权限、参与记账权限按私有组织规则来制定。私有链的应用场景一般是企业内部的应用，如数据库管理、审计等。也有一些比较特殊的组织情况，比如在政府行业的一些应用：政府的预算和执行，或者政府的行业统计数据，这个一般来说由政府登记，但公众有权力监督。私有链的价值主要是提供安全、可追溯、不可篡改、自动执行的运算平台，可以同时防范来自内部和外部对数据的安全攻击，这个在传统的系统是很难做到的。根据资料［1］的解读，央行发行数字货币可能就是一种私有链。和联盟链类似，私有链也是一种许可链。

币科学（Coin Science）公司推出供企业建立私链的多链（Multichain）平台。它提供保护隐私和权限控制的区块链平台，来克服在金融行业里碰到的推广区块链技术的障碍。多链的目标有以下3个：

1）保证区块链上的活动只能由选择的参与者看到；

2）引入机制来控制哪些交易是被允许的交易；

3）提供安全的挖矿机制，同时不需要工作量证明以及与其相关的成本。

多链把挖矿权限制在一组实名的矿工范围，解决了一直困扰私有链解决方案中的一方垄断挖矿过程的问题。它的解决办法是限制在同一个时间窗口同一矿工能产生的区块

链数。不像比特币那样只支持一条区块链，多链可以方便地配置多条区块链，并让用户同时用多条链。这样的话，机构用户可以让管理员配置区块链而不需要由区块链专业开发者来做。

多链让用户在一个配置文件中配置区块链的所有参数，这些参数包括：

- 区块链的协议，例如是私有链还是像比特币那样的公共链；
- 目标区块产生时间，例如 1 分钟；
- 权限，例如所有人能连接，只有一些人能发送或接收交易；
- 挖矿的不同形式（只适合于私有链）；
- 建立、移除管理员和矿工所需要的共识的程度，以及在建立期不需要强制执行的期限（只适合于私有链）；
- 矿工的报酬，例如每区块 50 个币，然后每 210 000 个区块减半付酬；
- 邻节点连接和 JSON RPC API 的 IP 端口，例如 8571、8570；
- 允许的交易类型，例如 paytoaddress、paytomultisig、paytoscripthash 等；
- 最大的区块大小，例如 1MB；
- 每个交易的最大元数据（OP_RETURN），例如 4KB。

多链在节点的“握手”连接过程如下：

1）每个节点提供它的公共地址，使其他节点能将它的地址包括在允许连接的清单中；

2）每个节点验证邻节点的地址是在它的授权连接的节点清单里；

3）每个节点发一个盘问（Challenge）消息给其他节点；

4）每个节点发回一个回复盘问信息的签名，证明拥有他们的对应公共地址的私钥；

5）如果双方对对方回复不满意，可随时中断连接。

在多链里，所有的权限的授予和回收都是通过包含特殊元数据的网络交易来实现的。找到创世区块的矿工被自动授予所有的权限，包括管理其他用户的管理员权限。管理员通过发交易给其他用户，并在交易的输出中包含授权用户的地址以及授权信息的元数据来给其他用户授予相应的权限。当要改变其他用户的管理和挖矿权限的时候，一个额外的限制条件是要由现有的管理员投票来决定。这些管理员的投票需要登记在不同的交易中，只有当足够的共识形成之后才能通过改变。

多链在很多方面的设计是为了使得用户在私链和比特币区块链能够进行双向迁移。多链是基于比特币核心的一个分叉。所有的对比特币的代码改变都是本地化的改变。未来比特币的升级功能可以并入多链的本地代码。它基于比特币的协议、交易和区块链架

构，只是在握手协议上有所改变。其他的功能是通过元数据，同时改变交易和区块的验证规则来实现的。在接口方面与比特币完全兼容，所有的新功能通过新的命令来提供。它可以做成普通比特币网络的一个节点。

多链提供一个在企业内快速部署私链的解决方案。可以用于如去中心化交易所、数据库同步、货币结算、债券发行和 P2P 交易、消费行业积分奖励机制等场景。

4. 侧链

比特币主要是按其设计者中本聪的思想设计的一个虚拟货币系统，虽然很成功，但是其规则已经相对固定，很难在比特币上做大的修改，因为这些修改会引起分叉，影响现有的比特币用户。因此，要在比特币平台上做创新或扩展是比较困难的。一般来说，大部分代币系统是通过用比特币平台做基础，重构一条区块链，然后在上面使用新的规则发新的虚拟货币。这就是目前大部分代币的做法。然而这些代币系统要从无到有得到人们的价值认可是非常困难的，通常的办法是与比特币挂钩，相当于用比特币作为储备来发行代币，这样就可以完成代币的货币价值认可的过程。但随之而来的问题是，如何自动保障代币和比特币的挂钩呢？因为虚拟货币的一个特点就是价格波动非常大，一般人都不愿意持有波动大、流动性差的代币。一个直接的想法就是通过比特币平台和代币平台的整合来做到实时的挂钩。

2014 年，亚当·贝克（Adam Back）等作者发表了一篇论文，题目是《Enabling Blockchain Innovations with Pegged Sidechains》，中文意思是“用与比特币挂钩的侧链来提供区块链创新”。其核心观点是“比特币”的区块链在概念上独立于作为资产的比特币。他希望通过技术能支持在不同的区块链上转移资产，这样新的系统可以重用原先的比特币。他提出一个侧链（Side Chains）的概念。所谓侧链，就是能和比特币区块链交互，并与比特币挂钩的区块链。贝克列出了侧链的一些属性：

- 一个用户在一条链上的资产被转移到另一条链上后，还应该可以转移回到原先链上的同一用户名下。
- 资产转移应该没有对手卷款逃跑的风险，也就是不诚实的用户没能力阻碍资产转移的发生。
- 资产的转移必须是原子操作，也就是要么全发生，要么不发生。不应该出现丢失资产或欺诈性增加资产的情况。
- 侧链间应该有防火墙。一条侧链上的软件错误造成链上资产的丢失或增加不会影响另一条链上的资产的丢失或增加。
- 即使在资产的转移过程中发生区块链的重组，也不应出现问题。任何因区块链重

组造成的中断，应该局限在本条侧链上而不应影响其他区块链。通常侧链之间最好能相互独立，用户可以从其他链条提供数据。只有当存在明确的侧链的共识规则时才需要去检查另一条侧链来对其验证。

- 用户不应需要跟踪不经常使用的侧链。

比特币是大家公认的公共链，是很多代币的基础。但比特币的设计规则决定了比特币有一定的局限，例如平均每 10 分钟出一个区块，每个区块 1MB 大小限制，这使得大概每秒才能确认 7 笔交易，这种交易速度而在很多场景下不能满足业务需求。因此，通过侧链来提升效率，扩展比特币功能是一个非常有效的做法。比如，闪电网络把很多交易放在侧链，只有在做清算时才用上主链，这样一来可以极大地提升交易速率，又不会增加主链的存储负担。

5. 互联链

如图 1-2 所示，针对特定领域的应用可能会形成各自垂直领域的区块链，这些区块链会有互联互通的需求，这样这些区块链也会通过某种互联互通协议连接起来。与互联网一样，这种区块链上的互联互通就构成互联链，形成区块链全球网络。

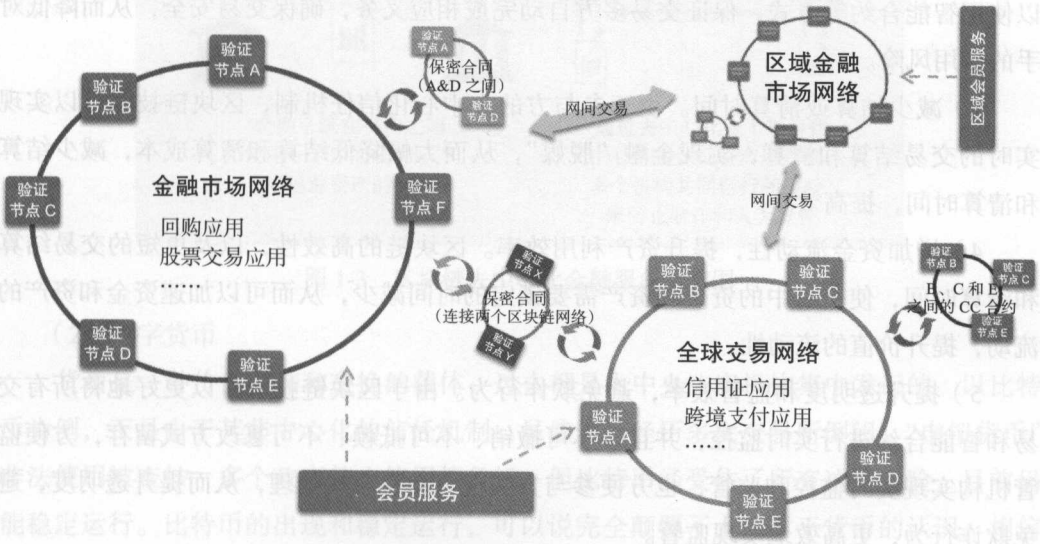


图 1-2　区块链网络示意图

1.1.4　区块链价值与应用

根据各个区块链采取的技术组合不同，形成的区块链特点也大不相同。但是需要指出的是，区块链技术是一揽子技术，可以根据业务的需要进行有针对性的组合和创新。

总体来说，去中心化信用机制是区块链技术的核心价值之一，因此区块链本身又被称为“分布式账本技术”“去中心化价值网络”等。自古以来，信用和信任机制就是金融和大部分经济活动的基础，随着移动互联网、大数据、物联网等信息技术的广泛应用，以及工业 4.0 等新一代工业革命的开启，网络空间的信用作为数字化社会的基石的作用显得更加重要。传统上，信用机制是中心化的，而中心化的信任和信用机制必然导致中心化机构成为价值链的核心，也容易引发问题。而区块链技术则首先在人类历史上实现了去中心化的大规模信用机制，在消除中心机构“超级信用”的同时，保证信用机制安全、高效地运行。

具体来看，区块链的颠覆性价值至少包括以下 5 个方面。

1）简化流程，提升效率。由于区块链技术是参与方之间通过共享共识的方式建立的公共账本，形成对网络状态的共识，因此区块链中的信息天然就是参与方认可的、唯一的、可溯源、不可篡改的信息源，因此原来许多重复验证的流程和操作就可以简化，甚至消除，例如银行间的对账、结算、清算等，从而大幅提升操作效率。

2）降低交易对手的信用风险。与传统交易需要信任交易对手不同，区块链技术可以使用智能合约等方式，保证交易多方自动完成相应义务，确保交易安全，从而降低对手的信用风险。

3）减少结算或清算时间。由于参与方的去中心化信任机制，区块链技术可以实现实时的交易结算和清算，实现金融“脱媒”，从而大幅降低结算和清算成本，减少结算和清算时间，提高效率。

4）增加资金流动性，提升资产利用效率。区块链的高效性，以及更短的交易结算和清算时间，使交易中的资金和资产需要锁定的时间减少，从而可以加速资金和资产的流动，提升价值的流动性。

5）提升透明度和监管效率，避免欺诈行为。由于区块链技术可以更好地将所有交易和智能合约进行实时监控，并且以不可撤销、不可抵赖、不可篡改方式留存，方便监管机构实现实时监控和监管，也方便参与方实现自动化合规处理，从而提升透明度，避免欺诈行为，更高效地实现监管。

区块链的创新性最大的特点不在于单点技术，而在于一揽子技术的组合，在于系统化的创新，在于思维的创新。而正是由于区块链是非常底层的、系统性的创新，区块链技术和云计算、大数据、人工智能、量子计算等新兴技术一起，被认为是最具变革性的新兴技术之一。其中，金融服务领域是即将被颠覆的关键领域之一，除此之外，区块链还可以被广泛应用于物联网、移动边缘计算等去中心化控制领域，以及智能化资产和共

享经济（如自动驾驶汽车、智能门锁 + 租赁）等一系列潜在可应用的领域。下面我们重点介绍几类区块链变革金融服务的场景。

（1）金融领域的结算和清算

以金融领域的结算和清算为例，全球每年涉及各种类型的金融交易高达 18 万亿美元。如图 1-3 所示，由于交易双方互不信任，因此金融机构需要通过处于中心位置的清算结构来完成资产清算和账本的确认。这类涉及多个交易主体且互不信任的应用场景就非常适合使用区块链技术。原则上，可以直接在金融之间构建联盟链，那么机构之间只需要共同维护同一个联盟区块链，即可实现资产的转移和交易。

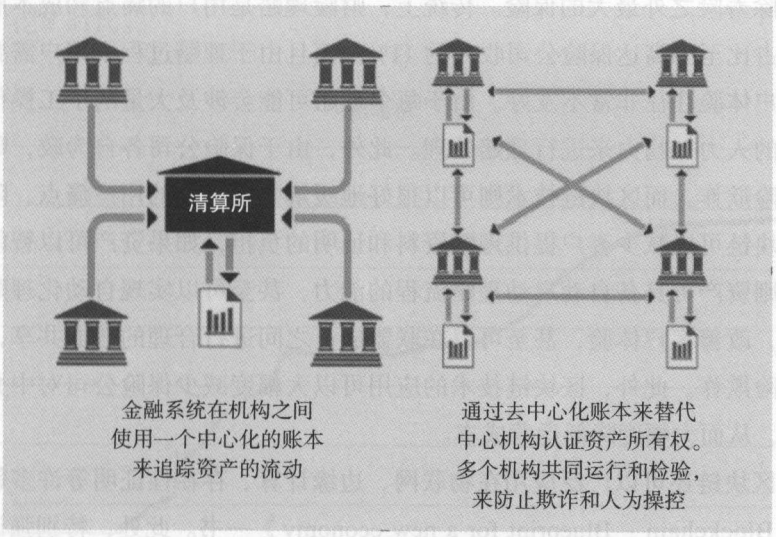


图 1-3　区块链去中心化金融服务示意图

（2）数字货币

货币是一种价值存储和交换的载体，过去都是由中央法定机构集中发行的。以比特币为例，正是由于其非中心化的信任机制，虽然先后经历多次交易所倒闭、“虚拟货币”非法使用被查抄、多个政府禁止使用等危机，但比特币经受住了所有这些考验，目前仍能稳定运行。比特币的出现和稳定运行，可以说完全颠覆了人们对于货币的认识。相信区块链技术或者说分布式账本技术会在数字货币技术体系中占据重要地位。

（3）跨境支付

另一个区块链可颠覆的金融服务就是跨境支付。通常跨境支付到账时间长达几天甚至一个星期。除此之外，跨境支付需要双边的用户都向当地银行提供大量开户资料和证明，以配合银行的合规性要求，参与交易的银行和中间金融机构还需要定期报告，以

实现反洗钱等其他合规性要求。这是一个典型的涉及多方主题的交易场景，区块链技术可以应用在多个环节。区块链技术，一方面可以减少用户重复提交证明材料，提升效率，另一方面可以更好地实现合规、实时性等，大幅提升金融机构的运行效率，提升监管效率。此外，由于区块链技术可以在银行等金融机构之间直接通过区块链实现资金和资产的转移，因此可以去掉高昂的中间费用。此外，还可以结合智能合约等技术，在合约中规定好实施支付的条件，在支付的同时保证义务的实施，提升交易的安全性。

（4）财产保险

财险是除寿险之外最大的保险。传统上，财险理赔是用户的痛点和成本瓶颈，估计理赔成本的占比至少高达保险公司收入的 11%。而且由于理赔过程中用户需要提供大量的资料，客户体验往往非常不友好。由于每个理赔可能会涉及大量的手工操作，因此需要占用大量的人力、物力来进行理赔处理。此外，由于保险公司各自为政，财险理赔还需要对抗保险欺诈。而区块链技术则可以很好地缓解财险理赔的用户痛点，降低理赔成本。首先区块链可以减少客户提供理赔资料和证明的负担，如果资产可以智能化地嵌入智能合约，则资产可具备自动启动理赔流程的能力，甚至可以实现自动化理赔，大幅加速理赔过程，改善客户体验，甚至可以在联盟成员之间进行合理的数据共享，有效地发现和排除保险欺诈。此外，区块链技术的应用可以大幅度减少保险公司对中介代理服务人员的需求，从而大幅度降低运营成本。

此外，区块链还可以广泛应用在物联网、边缘计算、存在性证明等许多领域，读者可以参考《Blockchain：Blueprint for a new economy》一书。此处，特别强调的是关于区块链的应用可能层出不穷，关键还是要理解区块链技术的内涵和变革原理，深刻体会区块链去中心化的系统化思维，从而可以结合自身对相关行业的理解和需求，创造出新的解决方案、新的价值。

1.2 区块链体验

区块链仍然是一个抽象概念，为了更好地理解区块链，为本书后续章节提供一个直观的理解基础，本节中我们将首先通过交易所购买少量比特币，然后转移到比特币钱包中，最后通过钱包实现比特币转账[⊖]。

⊖ 体验过程用到的现金可以转回交易所换回现金并提现，只会消耗少量的比特币作为矿工费用。

1.2.1　获取比特币的 3 种途径

获取比特币有 3 种途径：一是作为“矿工”挖矿获得，二是线上“交易所”购买或者线下通过中间人购买，三是作为商家收取比特币。

1. 挖矿

由于比特币的独特设计，参与者可以通过计算能力竞争的方式获取系统奖励和支付小费，同时也维护着比特币这个区块链的稳定运转，我们把这种算力竞争行为称为“挖矿”。比特币价格的一路攀升。挖矿的设备和算力也一路升级，如图 1-4 所示，从最初的 CPU 挖矿，到第二代的显卡挖矿，经历过短暂的 FPGA 挖矿时代后，迅速进入专用芯片（ASIC）挖矿时代。

图 1-4　比特币算力增长图

而进入 ASIC 矿机时代之后，矿机芯片的工艺升级速度远超摩尔定律的演进速度，差不多 3 个月时间就会进化一代，蚂蚁矿机 S9 是目前新出产的主流挖矿设备已经采用了 16nm 工艺制造的专用芯片。

“挖矿”今天已经成为高度专业化的细分产业。为保证收益，挖矿不仅要求有较高的初始投入，以及更低廉获取“矿机”和电力的渠道，还要求有专业的管理能力。如图 1-5 所示，这是一座位于我国西南某处的比特币矿场。

随着挖矿专业化程度的提高，矿工往往都是通过联合挖矿组成矿池的形式来挖矿的，矿池用来协调和分布挖矿的收益，比特币的算力分布目前前几大矿池都位于中国。

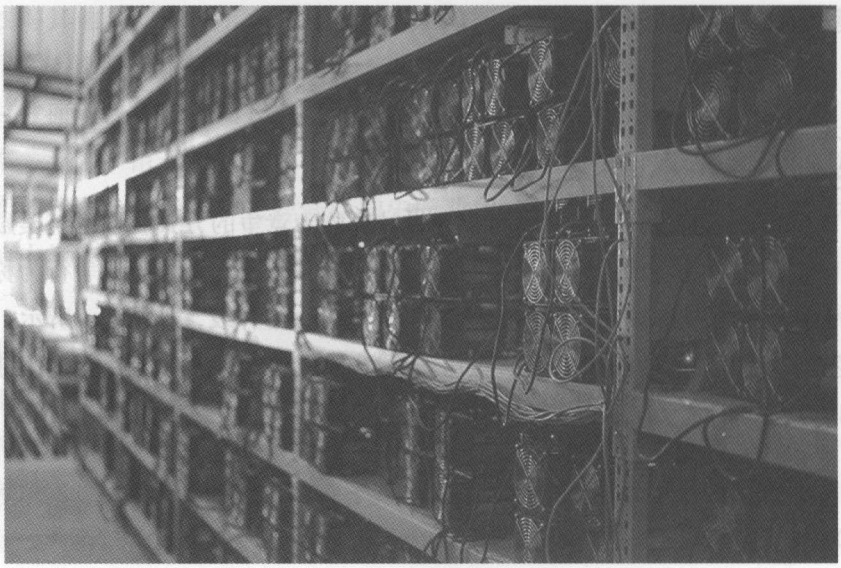


图 1-5 比特币矿场

2. 线上交易所或者线下撮合获取比特币

其中线上交易所方面，我国的 okcoin、火币占据了交易量的绝大多数，两家交易量占线上交易量的 93% 以上。线下交易具有更好的匿名性。图 1-6 展示的是比特币历史交易价格，可以看到从最初的不到 0.1 美元到历史最高点接近 1200 美元，再到当前日期（2016 年 7 月 25 日）的约 660美元。中间经历多次大的价格波动。

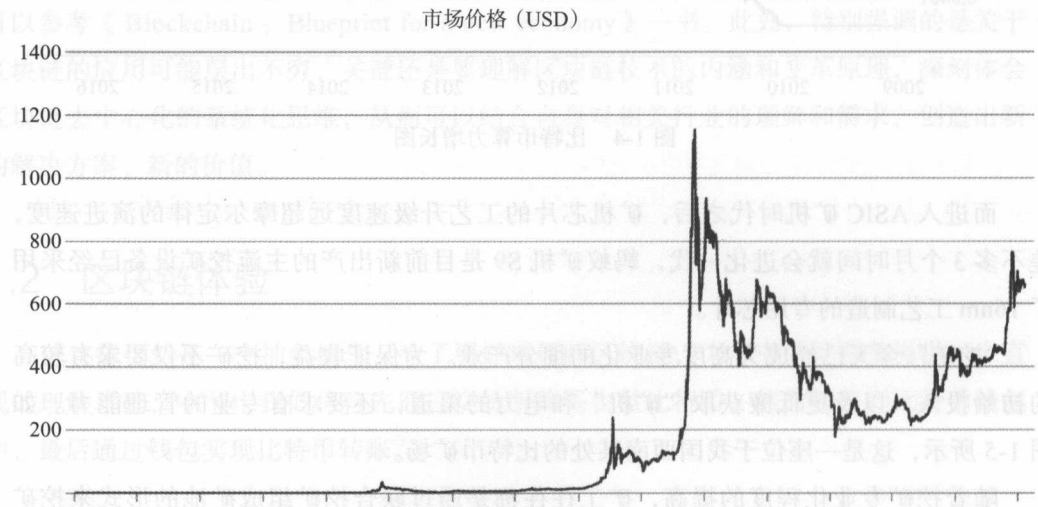


图 1-6 比特币历史价格（对数坐标，美元计价）

3. 比特币作为一种支付的手段

其优势在于跨境支付等场景下具备非常低的收费，并且非常快捷。在日常小额支付方面，目前在全球也有一定的市场。目前比特币作为一种支付手段，主要在欧美等发达国家和地区有比较广泛的分布。当然，由于比特币价格的波动性，一般商家都会实时将比特币转换为当地货币。比特币在我国不能作为货币支付手段，不能很方便地在银行汇兑。

1.2.2　通过交易所购买比特币

在本节中，我们将通过 OKCoin 这个比特币交易平台购买少量比特币。读者可以选择火币、BTCC 等其他平台购买获取比特币，基本过程是相似的。大部分主流交易平台也提供移动端 App，读者可以根据情况选用。

首先，我们需要注册 OKCoin 的账号，在 OKCoin 中国站（https://www.okcoin.cn/user/register.do）通过邮箱（或手机号）注册即可。如图 1-7 所示，填写邮箱、密码，并勾选同意服务条款后，单击“注册”按钮即可完成注册。

注册成功后可看到注册成功的页面，如图 1-8 所示。然后开始身份认证。根据相关条例要求，目前几乎所有比特币交易平台都会要求真实身份认证。

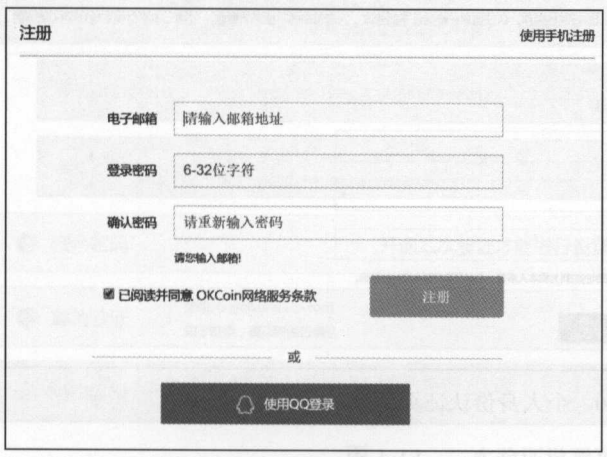


图 1-7　网站注册页面

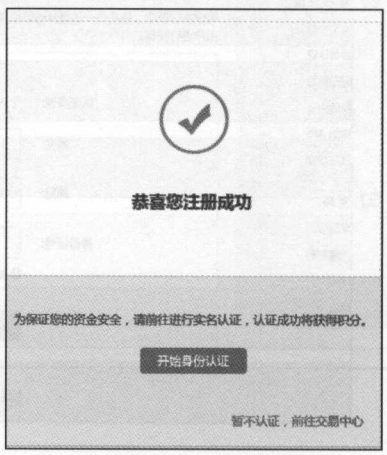


图 1-8　注册成功页面

单击图 1-8 中的“开始身份认证”按钮，将会进入如图 1-9 所示的提示页面，可以选择“个人用户”或者“企业用户”进行认证。这里选择“个人用户”这个类型进行认证。

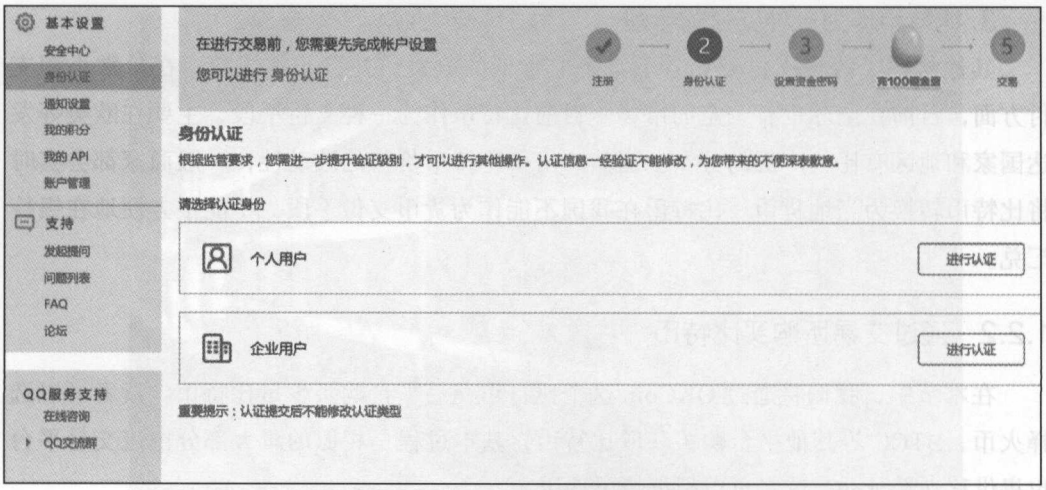


图 1-9　身份认证提示页面

如图 1-10 所示，正确填写身份信息并提交就能看到如图 1-11 所示的认证成功提示。注意，请使用真实身份信息，如遇到忘记密码等情形，可能会需要配合平台方提供相关证明才能进行处理。

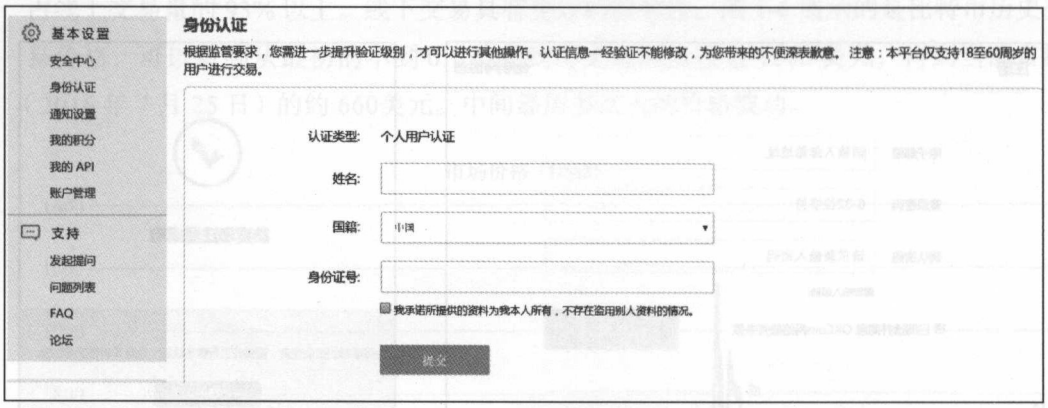


图 1-10　个人身份认证页面

单击“设置资金密码”按钮，就会进入如图 1-12 所示的页面。根据提示，我们可以选择手机认证或者 Google验证的方式来设置二次验证的方式。

我们选择 Google 验证的方式，安装 iOS 或者 Android 版 Google Authenticator 之后，单击图 1-13 中的“设置”按钮，打开 App，扫描左边的条形码后就能看到 OKCoin.cn 的动态密码了。将 App 中的动态密码输入弹出页面中，就能看到成功提示页面，同时也

可看到资金密码的“设置”按钮变为可用。单击该按钮将进入如图 1-13 所示的资金密码设置页面。

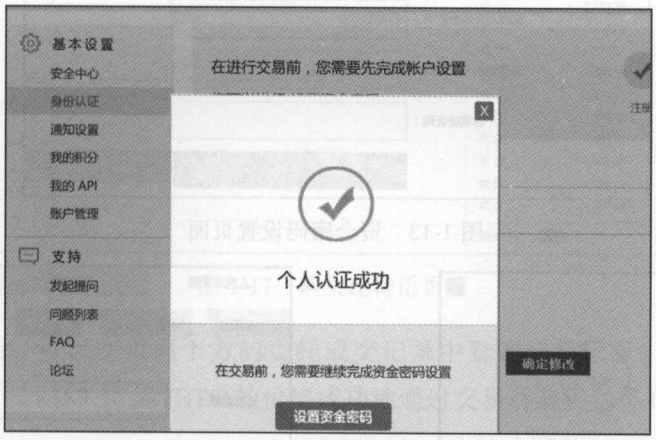


图 1-11　个人身份认证成功页面

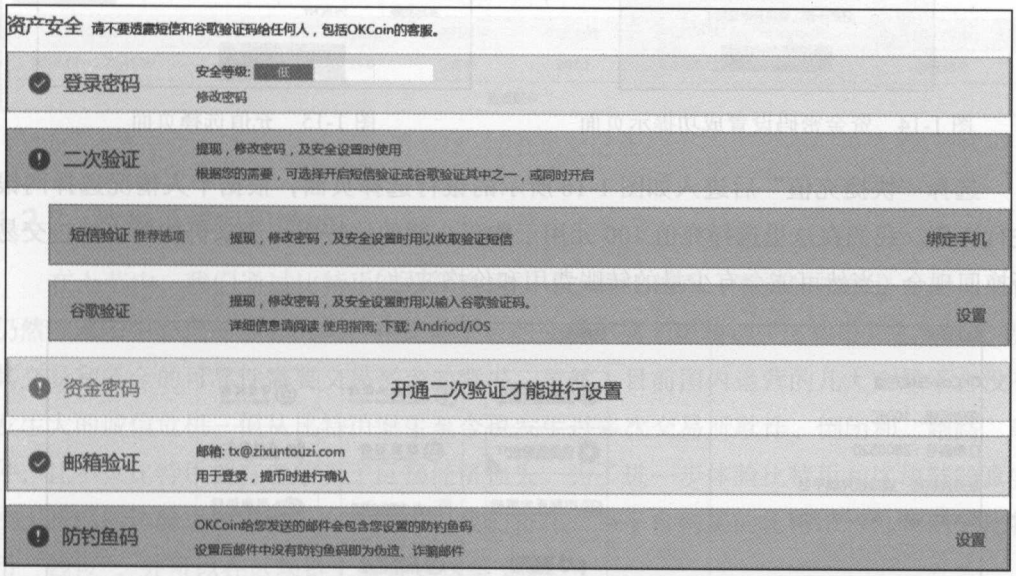


图 1-12　二次验证设置页面

设置密码并填写 Google 验证的二次验证密码（如果前面是手机验证，则是手机验证码），就会看到如图 1-14 所示的提示页面。

单击“前往充值”按钮进入充值页面，如图 1-15 所示。我们选择“快捷充值”方式，也可以选择“支付宝充值”或者“网银汇款充值”的方式。

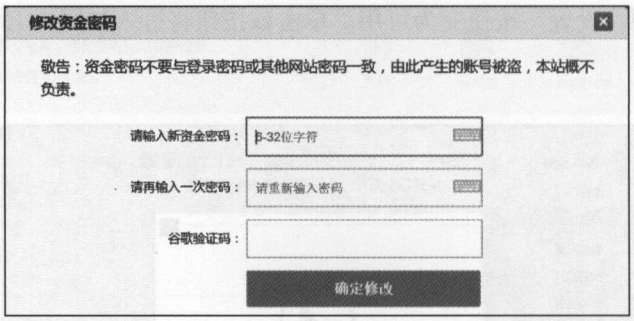


图 1-13 资金密码设置页面

图 1-14 资金密码设置成功提示页面

图 1-15 充值选择页面

选择“快捷充值”后进入如图 1-16 所示的银行选择页面，根据个人情况选择网银进行充值。我们在这里选择充值 100 元用于购买小额的比特币，未来仍然可以通过交易所换回现金（当然可能会有少量的转账费用和价格波动）。

图 1-16 快捷充值页面

充值成功之后就可以购买比特币了。我们可以通过“市价单”快速购买比特币，如图 1-17 所示。

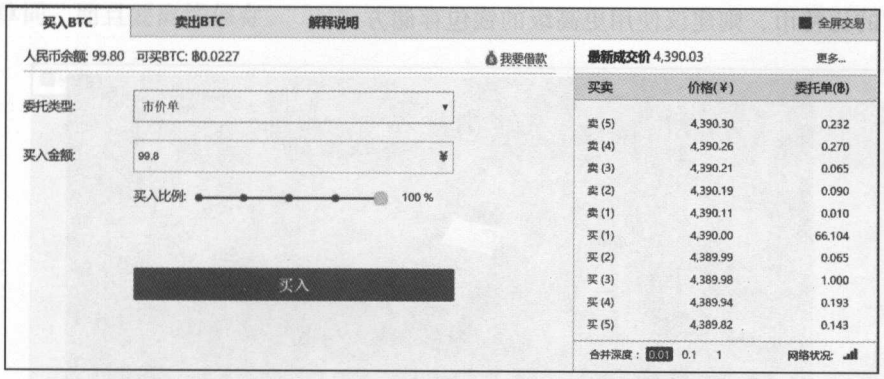


图 1-17　购买比特币页面

委托完成后，可以在页面下方的委托成交记录中看到交易记录，如图 1-18 所示。可以看到，我们以 4389.76 元 /BTC 的价格成功地通过交易所购买到了 0.02 个比特币。

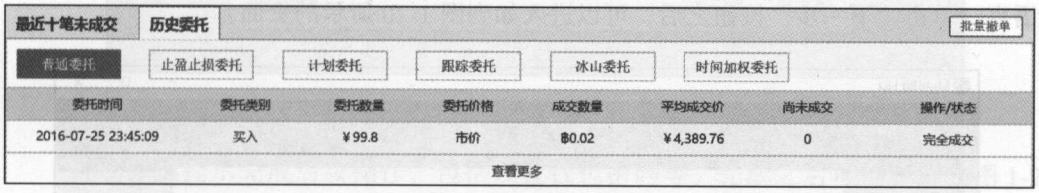


图 1-18　委托成交记录

1.2.3　比特币钱包和地址

在上节中，我们通过比特币交易平台购买了少量比特币。需要指出的是，交易平台仍然不属于中心化的服务机构，在交易平台的交易不属于区块链（比特币）之上的交易，其交易和资金的可靠性需要交易平台的背书。虽然，目前国内运营的几大交易平台没有发生大的诚信危机，但从比特币诞生至今也发生过多次交易所欺诈、倒闭和“跑路”事件，让不少比特币拥有者蒙受了巨额经济损失。为了进一步体验比特币和区块链的真实性，我们的体验之旅继续。在本节中，我们将在交易平台购买的比特币转入我们的比特币“钱包”，并可以在区块链上查询到这笔交易。

比特币钱包是一个形象的概念，比特币本身由一对数字密钥来决定归属，因为拥有私钥就能拥有对应地址比特币的处置权，可以说这些私钥就等于比特币，我们通常将管理这些数字密钥的软件称为“钱包”。比特币钱包，根据终端类型可以分为桌面钱包、手机钱包、网页钱包和硬件钱包。其中硬件钱包（见图 1-19）成本最高，也相对更安全。对于小量比特币来说，我们可以选用网页钱包这种轻量级的钱包来存储，而对于较

大额度的比特币，则建议使用更高级的钱包存储方式。

图 1-19 比特币硬件钱包 case（来源：choosecase.com）

我们接下来将选择开源钱包 MultiBit HD 桌面版，当然读者也可以选择其他优秀的钱包。在 https://multibit.org/ 下载对应版本的文件后，单击安装，并选择中文作为界面语言。单击“下一步”按钮之后，可以进入如到图 1-20 所示的页面。

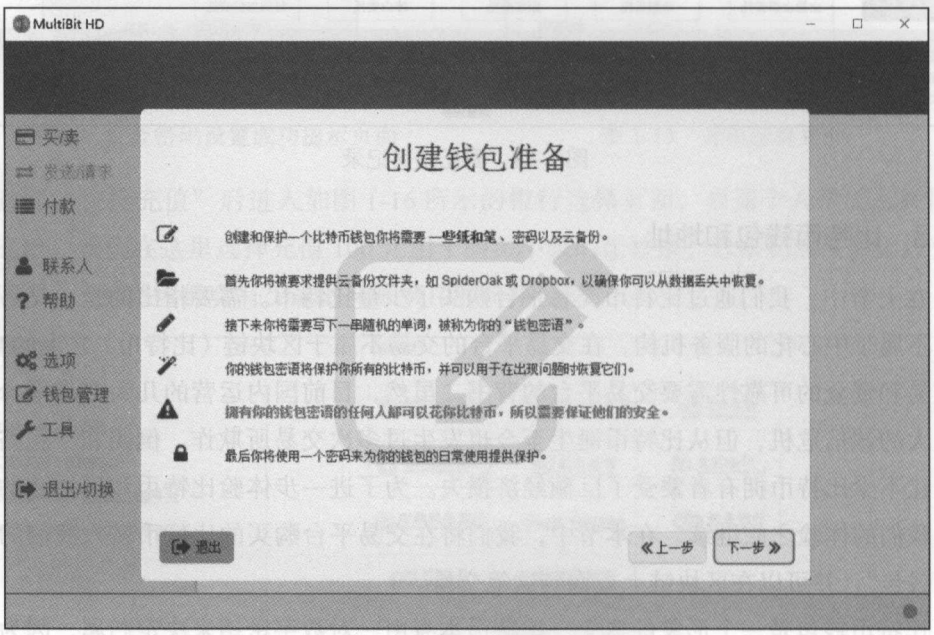


图 1-20 创建钱包准备页面

特别需要强调的是，比特币不同于银行账户的概念，钱包是帮助我们管理这些私钥的，同时也要妥善保管好钱包的恢复密语和备份数据。MulitBit HD 钱包使用一种新的密钥技术，即 12 个单词的密语可以恢复这个钱包，如图 1-21 所示。所以建议妥善保存

这些单词，而且要离线保存。

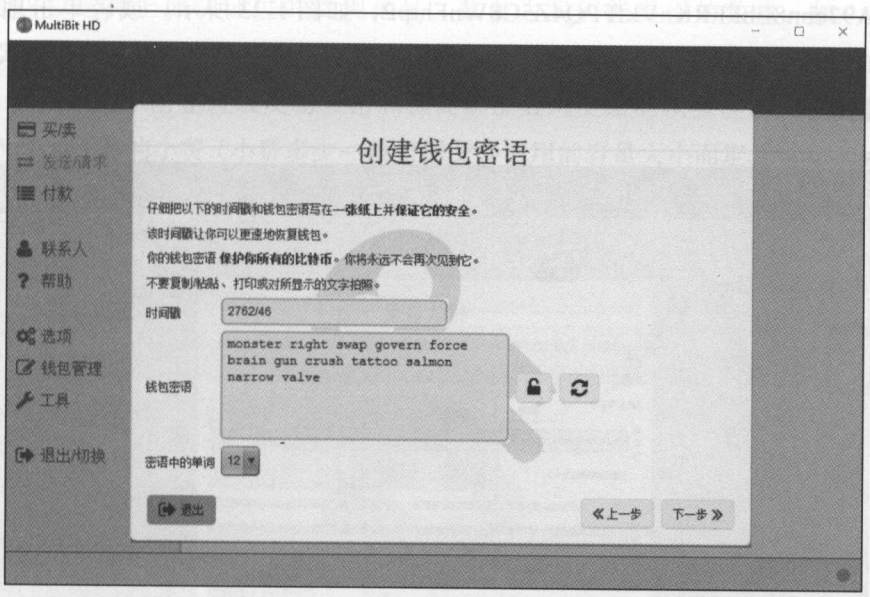


图 1-21　MultiBit 钱包密语

继续按照提示完成后续操作，包括设置钱包密码等。完成之后可以看到如图 1-22 所示的创建钱包报告页面。

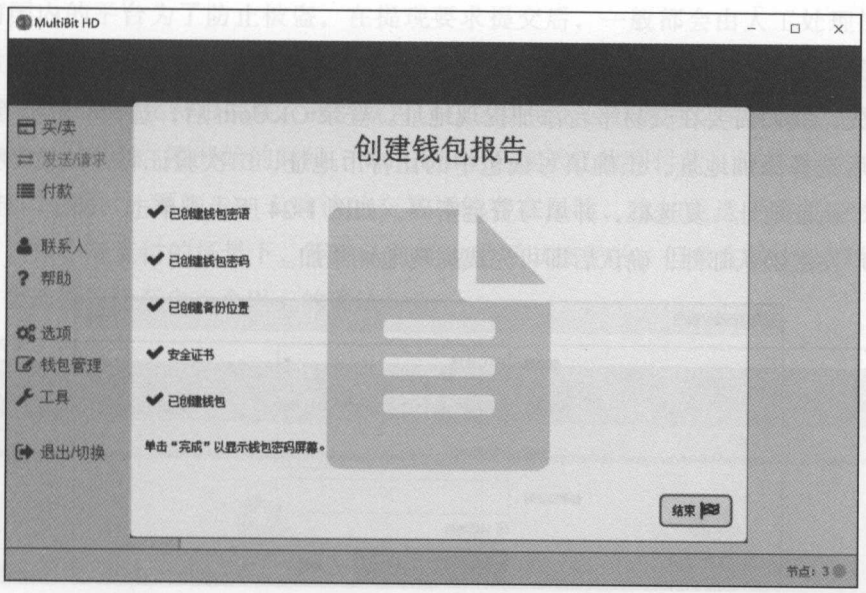


图 1-22　创建钱包报告页面

创建完成后打开 MultiBit，在发送 / 接受页面选择接收，可以看到钱包的比特币地址：1FA97cbn8EbFFRKnVkfFPQ4Z5C8WnFhtpP，如图 1-23 所示。或者单击地址栏后面第二个图标，可以显示二维码形式的比特币地址，这将是我们从交易平台购买的比特币提现地址。

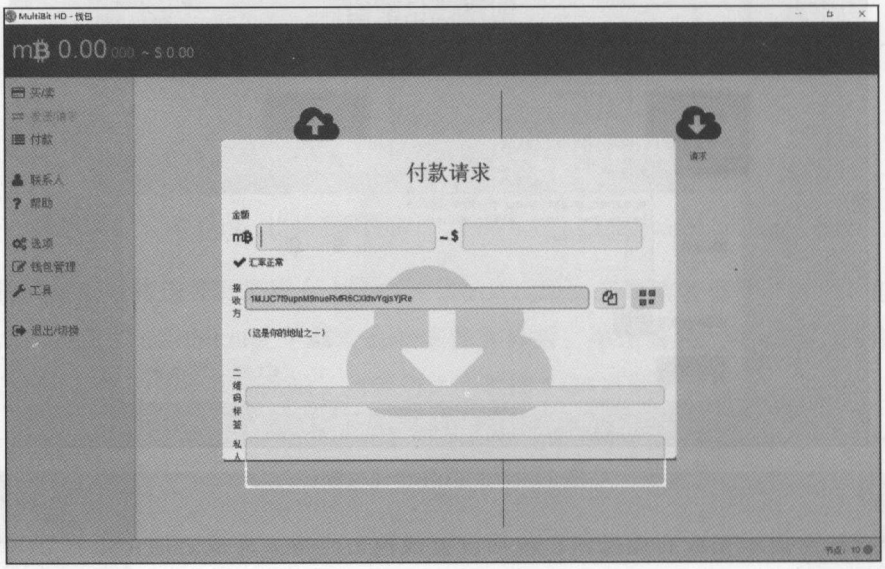


图 1-23　钱包比特币地址

1.2.4　从交易平台提取比特币到钱包

首先，我们需要在交易平台添加提现地址。登录 OKCoin 后，选择“资金管理”栏目，然后选择添加地址，正确填写钱包中的比特币地址，二次验证码，如果需要认证，则勾选“认证地址”复选框，并填写资金密码，如图 1-24 所示。单击“确定”后，平台会向用户发送确认邮件，确认后即可完成提现地址添加。

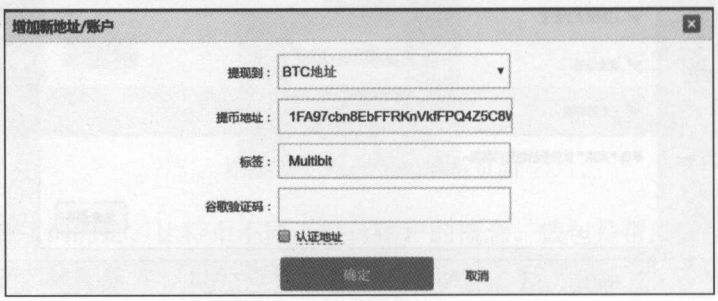


图 1-24　添加比特币提现地址

最后一步，在“资金管理”栏目中选择“BTC 提现”选项卡，如图 1-25 所示。提现地址可以选择上面认证过的提现地址，数量为我们能提现的数量，如 0.02BTC（20mBTC）。注意，“网络手续费”为网络“矿工”维持比特币区块链网络运转的交易费奖励。当然，为了防止垃圾交易攻击和提高矿工处理交易的积极性，一般都会选择 0.1 ~ 0.5mBT 不等的小费（小费多少一般根据交易占用的容量大小而定）。

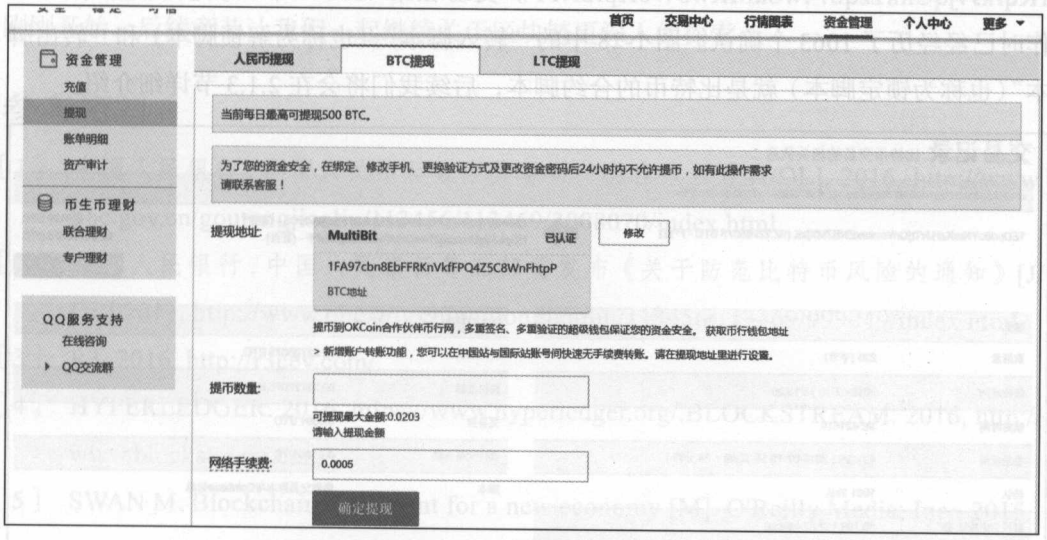


图 1-25　比特币提现页面

目前国内的平台为了防止被盗，在提现要求提交后，一般都会由人工处理提现申请，包括电话确认提现是本人所操作、确认提现的数量等，确认完成后才会正式处理。等平台将交易发送到比特币网络，我们就可以在区块链上公开看到这笔交易了。我们可以在 MultiBit 上看到，刚开始的时候，MultiBit 上会显示已收到付款，但是是“未确认”的，如图 1-26 所示。原则上，未确认的交易可能存在风险，比如发送者重复花费这部分比特币，在小额支付的场景下，零确认可能也是可以接受的，但是在较大金额的交易中，通常会选择等待至少 6 个以上的确认。

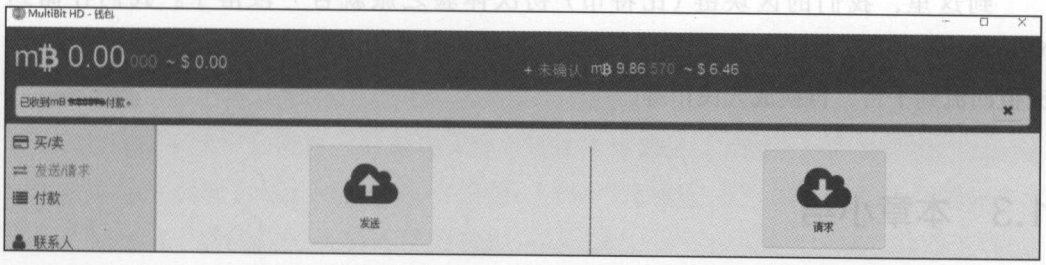


图 1-26　未确认收款

1.2.5 比特币交易查询

经过比较长的时间后，我们可以使用 blockchain.info 和 qukuai.com 查询交易的结果。如图 1-27 所示，这笔交易是从一个有 92.22788075 的 BTC，地址为 1EDpd8oYNmKzHJvTrjQnWmkexENB7MXjxK 中转出的，剩余的 92.20788075BTC 转到一个新地址 1KqrkJvjqUmrzzq274wSkMRwbWbXprkNPF。交易在第 421416 个区块被锁定，截至写作时已经经历了 1063 个确认。图 1-28 中的“转入脚本”（也称为解锁脚本）和“转出脚本”（也称为锁定脚本）就是比特币的合约脚本，后续我们将会在 2.1.3 节详细介绍。

图 1-27 BTC 提现交易结果

到这里，我们的区块链（比特币）初次体验之旅就告一段落了。我们存储到 MultiBit 钱包的比特币可以直接用于支付、捐赠、打赏，也可以通过交易平台的比特币充值回流到平台，再换成人民币等。

1.3 本章小结

本章中，我们首先简单介绍了区块链的起源和定义，以及区块链的分类、价值和应

用。然后我们通过图示的方式，以比特币这个目前最大的公链为例，带领大家体验比特币，包括如何获取比特币，如何通过交易平台购买比特币，以及如何通过钱包存储比特币，最后将交易平台的比特币提取到钱包中，并在区块链上公开查询到这笔交易。

毋庸置疑，区块链的发展已经远远超出了比特币和数字货币的范畴，可以说，区块链去中心化的信任机制和价值传递机制的变革将极具颠覆性，当前区块链领域的创新才刚刚开始。后续章节让我们一起继续关于区块链更深入的探索。

参考资料

[1] 中国人民银行．中国人民银行数字货币研讨会在京召开 [J/OL]. 2016, http://www.pbc.gov.cn/goutongjiaoliu/113456/113469/3008070/index.html.

[2] 中国人民银行．中国人民银行等五部委发布《关于防范比特币风险的通知》[J/OL].2013, http://www.pbc.gov.cn/goutongjiaoliu/113456/113469/999049/index.html.

[3] R3. 2016, http://r3cev.com/.

[4] HYPERLEDGER. 2016, https://www.hyperledger.org/.BLOCKSTREAM. 2016, http://www.blockstream.com/.

[5] SWAN M. Blockchain: Blueprint for a new economy [M]. O'Reilly Media, Inc., 2015.

第2章

区块链基础

区块链是随着比特币等数字加密货币的日益普及而逐渐兴起的一种全新技术，它提供了一种去中心化的、无需信任积累的信用建立范式，目前已经引起金融行业、科研机构、政府部门和投资公司的高度重视与广泛关注。区块链技术通过建立一个共同维护且不可被篡改的数据库来记录过去的所有交易记录和历史数据，所有的数据都是分布式存储且公开透明的。在这种技术下，任何互不相识的网络用户都可以通过合约、点对点记账、数字加密等方式达成信用共识，而不需要任何的中央信任机构。在这种技术下，我们可以建立数字货币、数字资产、智能财产以及智能合约等。

通过上一章的介绍，相信大家已经对区块链和比特币有了初步的认识，在本章中，我们将继续探讨区块链的技术细节。

本章将首先介绍区块链的相关基本概念及其运作原理，然后介绍区块链上可以进行的操作和相关细节，最后再讨论区块链上的交易流程和它的验证过程。

2.1 区块链技术

区块链本质上是一个对等网络（peer-to-peer）的分布式账本数据库。比特币的底层就采用了区块链的技术架构。区块链本身其实是一串链接的数据区块，其链接指针是采用密码学哈希算法对区块头进行处理所产生的区块头哈希值。每一个数据块中记录了一组采用哈希算法组成的树状交易状态信息，这样保证了每个区块内的交易数据不可篡改，区块链里链接的区块也不可篡改。

2.1.1　基本概念

一个完整的区块链系统包含了很多技术，其中有存储数据的数据区块及其之上的数字签名、时间戳等技术，有作为支撑的 P2P 网络和维护系统的共识算法，有挖矿和工作量证明机制，有匿名交易机制和比特币钱包，还有链龄、UTXO、Merkle 树、双花等相关技术概念。正是这些技术，使得区块链在无中心的网络上形成了运转不息的引擎，为区块链的交易、验证、链接等功能提供了源源不断的动力。

1. 数据区块

比特币的交易记录会保存在数据区块之中，比特币系统中大约每 10 分钟会产生一个区块，每个数据区块一般包含区块头（Header）和区块体（Body）两部分，如图 2-1 所示。

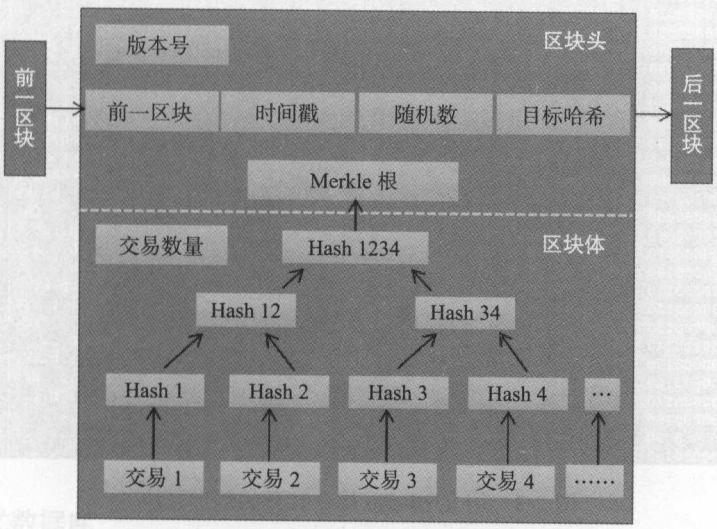


图 2-1　区块结构

区块头封装了当前的版本号（Version）、前一区块地址（Prev-block）、时间戳（Timestamp）、随机数（Nonce）、当前区块的目标哈希值（Bits）、Merkle 树的根值（Merkle-root）等信息。

区块体中则主要包含交易计数和交易详情。交易详情就是比特币系统中的记账本，每一笔交易都会被永久地记入数据区块中，而且任何人都可以查询。区块体中的 Merkle 树将会对每一笔交易进行数字签名，如此可以确保每一笔交易都不可伪造且没有重复交易。所有的交易将通过 Merkle 树的 Hash 过程产生一个唯一 Merkle 根值记入区块头。关于 Merkle 树本章后面将详细介绍。

如果你使用的是比特币核心钱包（Bitcoin core），那么每当你打开客户端时，区块数据文件都会被同步到电脑硬盘中，可以在 blocks 文件夹下找到它们。如图 2-2 所示的 .dat 文件就是我们要找的数据区块文件。

图 2-2 数据区块文件

我们还可以使用 hexdump 指令在终端上将数据区块以十六进制的方式显示出来。我们可以通过解析这些数据得出交易记录、区块大小等基本信息，因此我们说区块链中的数据是完全公开透明的。如图 2-3 所示，我们使用指令 hexdump -n 10000 -C blk00000.dat 打开了编号为 00000 的创世区块（比特币中的第一块区块链）。

```
00000000  f9 be b4 d9 1d 01 00 00  01 00 00 00 00 00 00 00  |................|
00000010  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000020  00 00 00 00 00 00 00 00  00 00 00 00 3b a3 ed fd  |............;...|
00000030  7a 7b 12 b2 7a c7 2c 3e  67 76 8f 61 7f c8 1b c3  |z{..z.,>gv.a....|
00000040  88 8a 51 32 3a 9f b8 aa  4b 1e 5e 4a 29 ab 5f 49  |..Q2:...K.^J)._I|
00000050  ff ff 00 1d 1d ac 2b 7c  01 01 00 00 00 01 00 00  |......+|........|
00000060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 ff ff  |................|
00000080  ff ff 4d 04 ff ff 00 1d  01 04 45 54 68 65 20 54  |..M.......EThe T|
00000090  69 6d 65 73 20 30 33 2f  4a 61 6e 2f 32 30 30 39  |imes 03/Jan/2009|
000000a0  20 43 68 61 6e 63 65 6c  6c 6f 72 20 6f 6e 20 62  | Chancellor on b|
000000b0  72 69 6e 6b 20 6f 66 20  73 65 63 6f 6e 64 20 62  |rink of second b|
000000c0  61 69 6c 6f 75 74 20 66  6f 72 20 62 61 6e 6b 73  |ailout for banks|
000000d0  ff ff ff ff 01 00 f2 05  2a 01 00 00 00 43 41 04  |........*....CA.|
000000e0  67 8a fd b0 fe 55 48 27  19 67 f1 a6 71 30 b7 10  |g....UH'.g..q0..|
000000f0  5c d6 a8 28 e0 39 09 a6  79 62 e0 ea 1f 61 de b6  |\..(.9..yb...a..|
00000100  49 f6 bc 3f 4c ef 38 c4  f3 55 04 e5 1e c1 12 de  |I..?L.8..U......|
00000110  5c 38 4d f7 ba 0b 8d 57  8a 4c 70 2b 6b f1 1d 5f  |\8M....W.Lp+k..._|
00000120  ac 00 00 00 00 f9 be b4  d9 d7 00 00 00 01 00 00  |................|
00000130  00 6f e2 8c 0a b6 f1 b3  72 c1 a6 a2 46 ae 63 f7  |.o......r...F.c.|
00000140  4f 93 1e 83 65 e1 5a 08  9c 68 d6 19 00 00 00 00  |O...e.Z..h......|
00000150  00 98 20 51 fd 1e 4b a7  44 bb be 68 0e 1f ee 14  |.. Q..K.D..h....|
00000160  67 7b a1 a3 c3 54 0b f7  b1 cd b6 06 e8 57 23 3e  |g{...T.......W#>|
00000170  0e 61 bc 66 49 ff ff 00  1d 01 e3 62 99 01 01 00  |.a.fI......b....|
00000180  00 00 01 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000190  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000001a0  00 00 00 ff ff ff ff 07  04 ff ff 00 1d 01 04 ff  |................|
```

图 2-3 用 hexdump 指令打开的创世区块

2. 挖矿与分叉问题

区块在挖矿过程中产生。所谓挖矿，实际上是穷举随机数算法，把上个区块的哈希值加上 10 分钟内的全部交易单打包，再加上一个随机数，算出一个 256 位的字符串哈希值，输入的随机数 Nonce 使哈希值满足一定条件就获得这个区块的交易记账权。新产生的区块需要快速广播出去，以便其他节点进行对其验证，以防造假。每个区块存着上一个区块的哈希值，可以溯源到源头，只有经过验证后才最终获得区块的交易记账权。比特币系统会让挖矿的矿工竞争记账权（在主链上链接区块的权利），这个竞争机制就是工作量证明机制。挖矿需要付出大量的能源和时间，谁付出的工作量多就能以更大的概率获得一个区块的记账权。获取记账权的矿工会将当前区块链接到前一区块，形成最新

的区块主链，该矿工也会得到系统奖励的一定数量（2009 ~ 2013 年每 10 钟产生 50 个比特币，2014 年至今每 10 分钟产生的比特币将减半成 25 个）的比特币。所有的区块链接在一起形成了区块链的主链，从创世区块到当前区块，在区块链之上的所有数据历史都可以被追溯和查询。

需要说明的是，可能会出现不同地区的两个矿工同时“挖出”两个新区块加以链接的情况，这时主链上就会出现“分叉”。系统并不会马上确认哪个区块不合理，而是约定后续矿工总是选择累计工作量证明最大的区块链。因此，当主链分叉以后，后续区块的矿工将通过计算和比较，将其区块链接到当前累计工作量证明最大化的备选链上，形成更长的新主链，并自动抛弃分叉处的短链，从而解决分叉问题。

3. 时间戳和不可篡改性

时间戳是指从格林威治时间 1970 年 01 月 01 日 00 时 00 分 00 秒（北京时间 1970 年 01 月 01 日 08 时 00 分 00 秒）起至现在的总秒数，通常是一个字符序列，唯一地标识某一刻的时间。在比特币系统中，获得记账权的节点在链接区块时需要在区块头中加盖时间戳，用于记录当前区块数据的写入时间。每一个随后区块中的时间戳都会对前一个时间戳进行增强，形成一个时间递增的链条。时间戳技术本身并没有多复杂，但在区块链技术中应用时间戳却是一个重大创新，时间戳为未来基于区块链的互联网和大数据增加了一个时间维度，使得数据更容易追溯，重现历史也成为可能。同时，时间戳可以作为存在性证明（Proof of Existence）的重要参数，它能够证实特定数据必然在某特定时刻是的确存在的，这保证了区块链数据库是不可篡改和不可伪造的，这也为区块链技术应用于公证、知识产权注册等时间敏感领域提供了可能。

4. 分布式数据库

比特币系统中的区块就像一个记账本一样，记录了所有比特币的交易信息，每一个比特币用户的比特币收支情况都被永久地嵌入了数据区块中以供别人查询。这些数据区块中的交易数据存放在每一个比特币用户的客户端节点中，所有的这些节点则组成了比特币及其坚韧的分布式数据库系统。任何一个节点的数据被破坏都不会影响整个数据库的正常运转，因为其他的健康节点中都保存了完整的数据库。

5. UTXO 交易模式

UTXO（Unspent Transaction Outputs）是未花费的交易输出，它是比特币交易过程中的基本单位。除创世区块以外，所有区块中的交易（Tx）会存在若干个输入（Tx_in，也称资金来源）和若干个输出（Tx_out，也称资金去向），创世区块和后来挖矿产生的区

块中给矿工奖励的交易没有输入，除此之外，在比特币系统中，某笔交易的输入必须是另一笔交易未被使用的输出，同时这笔输入也需要上一笔输出地址所对应的私钥进行签名。当前整个区块链网络中的 UTXO 会被储存在每个节点中，只有满足了来源于 UTXO 和数字签名条件的交易才是合法的。所以区块链系统中的新交易并不需要追溯整个交易历史，就可以确认当前交易是否合法。

6. 哈希函数

哈希函数在比特币系统中也有着重要的应用，区块链中的数据并不只是原始数据或者交易记录，还包括它们的哈希函数值，即将原始数据编码为特定长度的、由数字和字母组成的字符串后，记入区块链。哈希函数有着很多适合存储区块链数据的优点：

1）哈希函数处理过的数据是单向性的，通过处理过的输出值几乎不可能计算出原始的输入值；

2）哈希函数处理不同长度的数据所耗费的时间是一致的，输出值也是定长的；

3）哈希函数的输入值即使只相差一个字节，输出值的结果也会迥然不同。比特币系统中最常采用的哈希函数是双 SHA256 哈希函数，通俗来说就是将不同长度的原始数据用两次 SHA256 哈希函数进行处理，再输出长度为 256 的二进制数字来进行统一的识别和存储。

总之，哈希函数是比特币系统中的关键技术，为比特币系统提供了很多便利。本书后面的章节将会对哈希函数做详细介绍，此处不赘述。

7. Merkle 树

Merkle 树是数据结构中的一种树，可以是二叉树，也可以是多叉树，它具有树结构的所有特点。如图 2-4 所示，比特币区块链系统中的采用的是 Merkle 二叉树，它的作用主要是快速归纳和校验区块数据的完整性，它会将区块链中的数据分组进行哈希运算，向上不断递归运算产生新的哈希节点，最终只剩下一个 Merkle 根存入区块头中，每个哈希节点总是包含两个相邻的数据块或其哈希值。在比特币系统中使用 Merkle 树有诸多优点：首先是极大地

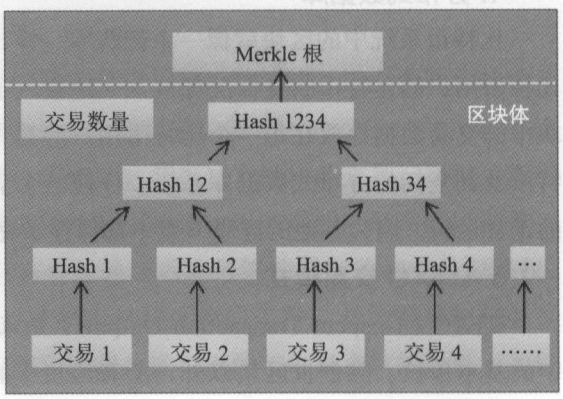


图 2-4　区块中的 Merkle 树

提高了区块链的运行效率和可扩展性，使得区块头只需包含根哈希值而不必封装所有底层数据，这使得哈希运算可以高效地运行在智能手机甚至物联网设备上；其次是 Merkle 树可支持“简化支付验证协议”（SPV），即在不运行完整区块链网络节点的情况下，也能够对交易数据进行检验。所以，在区块链中使用 Merkle 树这种数据结构是非常具有意义的。本书后面的章节将会对 Merkle 树做详细介绍。

8. 双重支付

双重支付问题又称为“双花”问题，即利用货币的数字特性用“同一笔钱”完成两次或者多次支付。在传统的金融和货币体系中，由于金钱货币是物理实体，具有客观唯一存在的属性，所以可以避免双重支付的情况。但在其他的电子货币系统中，则需要可信的第三方管理机构提供保证。区块链技术则在去中心化的系统中不借助任何第三方机构而只通过分布式节点之间的相互验证和共识机制，有效地解决了双重支付问题，在信息传输的同时完成了价值转移。区块链技术通过区块链接形成的时间戳技术加上验证比特币是否满足 UTXO（未花费交易）和数字签名，有效避免了双重支付的问题。如果有人用同一笔 UTXO 构造了两笔付给不同交易方的交易，则比特币客户端只会转发最先被侦听到的那个。矿工会选择将那笔交易包入未来区块，当其中一笔交易所在的区块后有 5 个链接的区块，这笔交易已经得到了 6 次确认。在比特币区块链上，6 次确认后可以基本上保证比特币不被双花。

9. P2P 网络

P2P 网络（peer-to-peer network，对等网络）是一种在对等者（peer）之间分配任务和工作负载的分布式应用架构，是对等计算模型在应用层形成的一种组网或网络形式。因此，从字面上，P2P 可以理解为对等计算或对等网络，P2P 网络示意图如图 2-5 所示。国内的迅雷软件采用的就是 P2P 技术。区块链系统是建立在 IP 通信协议和分布式网络的基础上的，它不依靠传统的电路交换，而是建立于网络通信之上，完全通过互联网去交换信息。网络中所有的节点具有同等的地位，不存在任何特殊化的中心节点和层级结构，每个节点均会承担网络路由、验证数据区块等功能。网络的节点根据存储数据量的不同可以分为全节点和轻量级节点，全节点存储了从创世区块以来的所有区块链数据（比特币网络现在大约有几十 GB，且还在不断增长中）。全节点的优点是进行数据校验时不需要依靠别的节点，仅依靠自身就可以完成校验更新等操作，缺点是硬件成本较高。轻量级节点只需要存储部分数据信息，当需要别的数据时可以通过简易支付验证方式（Simplified Payment Verification，SPV）向邻近节点请求所需数据来完成验证更新。

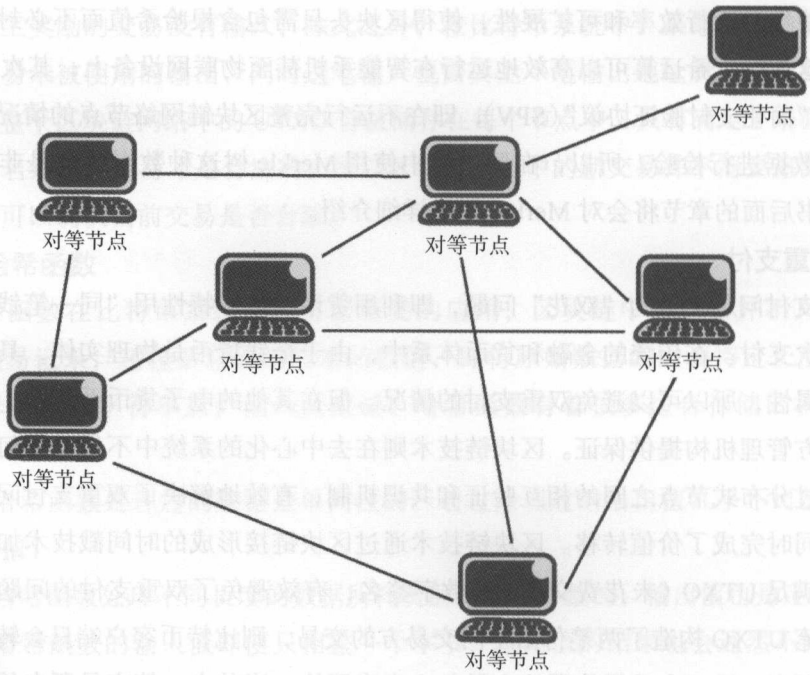


图 2-5 P2P 网络

10. 加密算法

除了哈希算法以外，比特币中还存在一种为交易加密的非对称加密算法（椭圆曲线加密算法）。非对称加密算法指的就是存在一对数学相关的密钥，使用其中一个密钥进行加密的数据信息，只有使用另一个密钥才能对该信息进行解密。这对密钥中，对外公开的密钥叫作公钥，不公开的密钥就叫作私钥。打个比方来说，公钥就像银行的账户，私钥就像是该账户的密码或者账户所有者的签名。区块链之上的有效交易有一个用于交易发起方私钥签名有效的数字签名，而该交易的签名可以通过使用交易发起方的公钥进行验证。公钥可以通过算法从私钥中计算得出，但私钥却不能从公钥中推出。比特币系统中使用的就是一种非常典型的非对称加密算法——椭圆曲线加密算法（ECC）。

如图 2-6 所示，比特币系统一般从操作系统底层的一个密码学安全的随机源中取出一个 256 位随机数作为私钥，私钥总数为 2^{256} 个，所以很难通过遍历所有可能的私钥得出与公钥的对应的私钥。用户使用的私钥还会通过 SHA256 和 Base58 转换成易书写和识别的 50 位长度的私钥，公钥则首先由私钥和 Secp256k1 椭圆曲线算法生成 65 字节长度的随机数。一般情况下，比特币钱包的地址也由公钥所生成，其生成过程为首先将公钥进行 SHA256 和 RIPEMD160 双哈希运算，并生成 20 字节长度的摘要结

果（即 Hash160 结果），这个将作为比特币地址的主体（body）信息，再在前面加上版本前缀 0x00，在后面添加 4 个字节的地址校验码。地址校验码通过对摘要结果进行两次 SHA256 运算，取哈希值的前 4 位产生。最后通过 Base58 处理把连在一起的版本前缀、主体信息和校验码转换成可以容易让人识别的比特币字符地址。

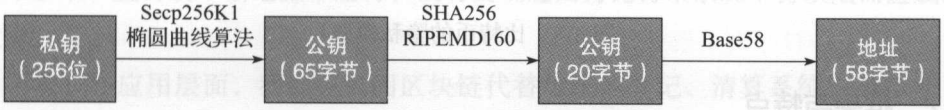


图 2-6　比特币非对称加密机制

11. 数字签名

数字签名就是在信息后面加上另一段内容，作为发送者的证明并且证明信息没有被篡改。一般是发送者将信息用哈希算法处理得出一个哈希值，然后用私钥对该哈希值进行加密，得出一个签名。然后发送者再将信息和签名一起发送给接收者。接收者使用发送者的公钥对签名进行解密，还原出哈希值，再通过哈希算法来验证信息的哈希值和解密签名还原出来的哈希值是否一致，从而可以鉴定信息是否来自发送者或验证信息是否被篡改。

12. 比特币的隐私模型

传统隐私模型（见图 2-7）为交易的参与者提供了一定程度的隐私保护，第三方不会交出交易者的个人身份信息，公众所得知的只是某个人将一定数量的货币发给了另外一个人，但是难以将该交易与某个特定身份的人联系起来，公众无法知道这人到底是谁。这同股票交易所发布的信息是类似的，每一手股票买卖发生的时间、交易量是记录在案且可供查询的，但是交易双方的身份信息却不予透露。但实际上，交易双方的个人信息是存放在第三方机构，所以一定程度上交易参与者的隐私信息还是会有泄露的风险。

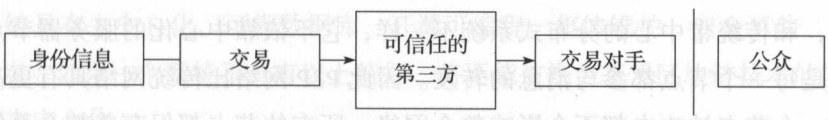


图 2-7　传统隐私模型

在比特币的隐私模型（见图 2-8）中，所有的交易不需要第三方的操控，也不需要提供任何身份信息，只需要提供比特币的地址就可以跟任何人完成一次准匿名的交易。在一定程度上，交易不可追溯到交易者本身，因此比特币上的交易可以在一定程度上摆脱监管。但通过对区块链上交易的地址以及交易额做关联分析，也可以获得有关交易

者的蛛丝马迹。因此，比特币的交易还不是纯粹的匿名交易机制，而是准匿名（pseudo-anonymous）交易机制。

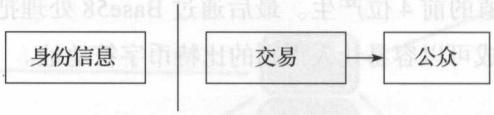


图 2-8 比特币的隐私模型

2.1.2 框架与特点

1. 框架简介

目前大多数区块链技术的应用与比特币类似，大部分是在比特币架构基础上的扩展。目前，区块链技术在金融行业得到广泛关注，被认为可以用来从最底层重构传统金融业现有的 IT 基础架构。我们将区块链的基础架构分为三层来进行讲解，如图 2-9 所示。

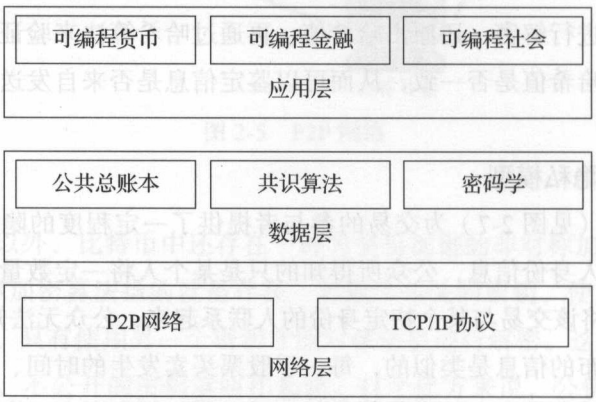


图 2-9 区块链基础架构

首先，在网络层之上，区块链是建立在 IP 通信协议和对等网络的基础上的一个分布式系统，和传统带中心的分布式系统不一样，它不依靠中心化的服务器节点来转发消息，而是每一个节点都参与消息的转发。因此 P2P 网络比传统网络具有更高的安全性，任何一个节点被攻击都不会影响整个网络，所有的节点都保存着整个系统的状态信息。

其次，在数据层面上，区块链就是一个只可追加、不可更改的分布式数据库系统，是一个分布式账本。如果是公开的区块链，也就是公有链，那么这个账本可以被任何人在任何地方进行查询，完全公开透明。在区块链网络中，节点通过使用共识算法来维持网络中账本数据库的一致性。同时采用密码学的签名和哈希算法来确保这个数据库不可

篡改，不能作伪，并且可追溯。例如，在比特币系统中，只有在控制了 51% 的网络算力时才有可能对区块链进行重组以修改账本信息。由于比特币系统的设计者中本聪在系统设计中巧妙地加入了带有经济激励的挖矿工作量证明（PoW）机制，使得即使拥有网络 51% 以上算力的人也不会损害其自身利益而发起对网络的攻击。因此，比特币系统自上线 7 年多来一直持续不断地正常运行，没有出现过因为比特币系统本身缺陷而造成的安全故障。

再次，在应用层面，我们可以用区块链代替传统的登记、清算系统。2016 年 6 月 22 日，波士顿咨询公司指出，到 2030 年，全球支付业务收入预计将会达到 8070 亿美元。基于区块链技术的汇兑和支付属于区块链的 1.0 应用版，其安全性、交易时间、成本都会对传统支付业务进行颠覆式改进。花旗银行也明确指出，到 2020 年，如果各大金融机构都使用区块链技术，每年能够节省超过 200 亿美元的成本。国信证券分析报告指出，通过区块链的点对点分布式的时间戳服务器来生成依照时间前后排列并加以记录的电子交易证明，可以解决双重支付问题，从而带来结算成本趋零的可能性。根据德国银行的一份引用波士顿咨询的研究报告，欧洲银行的 IT 成本支出平均占据银行整体运行成本的 16%[5]。一个重要原因就是传统银行在账本的维护、支付交易的结算和清算方面的架构过于复杂，维护成本过高。

在应用方面，区块链平台能够提供编程环境让用户编写智能合约。通过智能合约，可以把业务规则转化成在区块链平台自动执行的合约，该合约的执行不依赖可信任的第三方，也不受人为的干预。理论上只要一旦部署，一旦符合合约执行的条件就会自动执行。执行结果也可以在区块链上供公开检查，提供了合约的公正性和透明性。因此，智能合约可以降低合约建立、执行和仲裁中所涉及的中间机构成本。区块链的智能合约奠定了未来建立可编程货币、可编程金融，甚至是可编程社会的基础。

2. 架构特点

区块链具有去中心化、可靠数据库、开源可编程、集体维护、安全可信、交易准匿名性等特点。如果一个系统不具有以上特征，将不能被视为基于区块链技术的应用。

（1）去中心化

区块链数据的存储、传输、验证等过程均基于分布式的系统结构，整个网络中不依赖一个没有中心化的硬件或管理机构。作为区块链一种部署模式，公共链网络中所有参与的节点都可以具有同等的权利和义务。

（2）可靠数据库

区块链系统的数据库采用分布式存储，任一参与节点都可以拥有一份完整的数据库

拷贝。除非能控制系统中超过一半以上的算力，否则在节点上对数据库的修改都将是无效的。参与系统的节点越多，数据库的安全性就越高。并且区块链数据的存储还带有时间戳，从而为数据添加了时间维度，具有极高的可追溯性。

（3）开源可编程

区块链系统通常是开源的，代码高度透明公共链的数据和程序对所有人公开，任何人都可以通过接口查询系统中的数据。并且区块链平台还提供灵活的脚本代码系统，支持用户创建高级的智能合约、货币和去中心化应用。例如，以太坊（Ethereum）平台即提供了图灵完备的脚本语言，供用户来构建任何可以精确定义的智能合约或交易类型。关于以太坊的更多内容请参考 2.2 节。

（4）集体维护

系统中的数据块由整个系统中所有具有记账功能的节点来共同维护，任一节点的损坏或失去都不会影响整个系统的运作。

（5）安全可信

区块链技术采用非对称密码学原理对交易进行签名，使得交易不能被伪造；同时利用哈希算法保证交易数据不能被轻易篡改，最后借助分布式系统各节点的工作量证明等共识算法形成强大的算力来抵御破坏者的攻击，保证区块链中的区块以及区块内的交易数据不可篡改和不可伪造，因此具有极高的安全性。

（6）准匿名性

区块链系统采用与用户公钥挂钩的地址来做用户标识，不需要传统的基于 PKI（Public Key Infrastructure）的第三方认证中心（Certificate Authority，CA）颁发数字证书来确认身份。通过在全网节点运行共识算法，建立网络中诚实节点对全网状态的共识，间接地建立了节点间的信任。用户只需要公开地址，不需要公开真实身份，而且同一个用户可以不断变换地址。因此，在区块链上的交易不和用户真实身份挂钩，只是和用户的地址挂钩，具有交易的准匿名性。。

区块链技术的核心优势是去中心化，能够通过运用哈希算法、数字签名、时间戳、分布式共识和经济激励等手段，在节点无需互相信任的分布式系统中建立信用，实现点对点交易和协作，从而为中心化机构普遍存在的高成本、低效率和数据存储不安全等问题提供了解决方案。近年来，伴随着国内外研究机构对区块链技术的研究与应用，区块链的应用前景受到各行各业的高度重视，被认为是继大型机、个人电脑、互联网、移动/社交网络之后计算范式的第 5 次颠覆式创新，是人类信用进化史上继血亲信用、贵金属信用、央行纸币信用之后的第 4 个里程碑。它被视为下一代云计算的雏形，有望彻底

重塑人类社会活动形态，并实现从现在的信息互联网到价值互联网的转变。

2.1.3　区块链运作的核心技术

1. 区块链的链接

顾名思义，区块链即由一个个区块组成的链。每个区块分为区块头和区块体（含交易数据）两个部分。区块头包括用来实现区块链接的前一区块的哈希（PrevHash）值（又称散列值）和用于计算挖矿难度的随机数（nonce）。前一区块的哈希值实际是上一个区块头部的哈希值，而计算随机数规则决定了哪个矿工可以获得记录区块的权力。区块链的链接模型如图 2-10 所示。

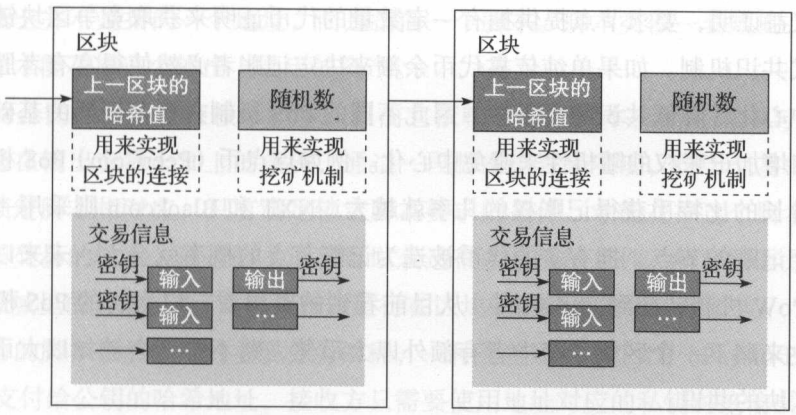


图 2-10　区块链的链接模型

2. 共识机制

区块链是伴随比特币诞生的，是比特币的基础技术架构。可以将区块链理解为一个基于互联网的去中心化记账系统。类似比特币这样的去中心化数字货币系统，要求在没有中心节点的情况下保证各个诚实节点记账的一致性，就需要区块链来完成。所以区块链技术的核心是在没有中心控制的情况下，在互相没有信任基础的个体之间就交易的合法性等达成共识的共识机制。

区块链的共识机制目前主要有 4 类：PoW、PoS、DPoS、分布式一致性算法。

（1）PoW

PoW（工作量证明），也就是像比特币的挖矿机制，矿工通过把网络尚未记录的现有交易打包到一个区块，然后不断遍历尝试来寻找一个随机数，使得新区块加上随机数的哈希值满足一定的难度条件，例如前面 10 位是零。找到满足条件的随机数，就相当于确定了区块链最新的一个区块，也相当于获得了区块链的本轮记账权。矿工把满足挖矿

难度条件的区块在网络中广播出去，全网其他节点在验证该区块满足挖矿难度条件，同时区块里的交易数据符合协议规范后，将各自把该区块链接到自己版本的区块链上，从而在全网形成对当前网络状态的共识。

- 优点：完全去中心化，节点自由进出，避免了建立和维护中心化信用机构的成本。只要网络破坏者的算力不超过网络总算力的 50%，网络的交易状态就能达成一致。
- 缺点：目前比特币挖矿造成大量的资源浪费；另外挖矿的激励机制也造成矿池算力的高度集中，背离了当初去中心化设计的初衷。更大的问题是 PoW 机制的共识达成的周期较长，每秒只能最多做 7 笔交易，不适合商业应用。

（2）PoS

PoS 权益证明，要求节点提供拥有一定数量的代币证明来获取竞争区块链记账权的一种分布式共识机制。如果单纯依靠代币余额来决定记账者必然使得富有者胜出，导致记账权的中心化，降低共识的公正性，因此不同的 PoS 机制在权益证明的基础上，采用不同方式来增加记账权的随机性来避免中心化。例如点点币（PeerCoin）PoS 机制中，拥有最多链龄长的比特币获得记账权的几率就越大。NXT 和 Blackcoin 则采用一个公式来预测下一个记账的节点。拥有多的代币被选为记账节点的概率就会大。未来以太坊也会从目前的 PoW 机制转换到 PoS 机制，从目前看到的资料看，以太坊的 PoS 机制将采用节点下赌注来赌下一个区块，赌中者有额外以太币奖，赌不中者会被扣以太币的方式来达成下一区块的共识。

- 优点：在一定程度上缩短了共识达成的时间，降低了 PoW 机制的资源浪费。
- 缺点：破坏者对网络攻击的成本低，网络的安全性有待验证。另外拥有代币数量大的节点获得记账权的几率更大，会使得网络的共识受少数富裕账户支配，从而失去公正性。

（3）DPoS

DPoS（股份授权证明）机制，类似于董事会投票。比特股（bitshares）采用的 PoS 机制是持股者投票选出一定数量的见证人，每个见证人按序有两秒的权限时间生成区块，若见证人在给定的时间片不能生成区块，区块生成权限交给下一个时间片对应的见证人。持股人可以随时通过投票更换这些见证人。DPoS 的这种设计使得区块的生成更为快速，也更加节能。

- 优点：大幅缩小参与验证和记账节点的数量，可以达到秒级的共识验证。
- 缺点：选举固定数量的见证人作为记账候选人有可能不适合于完全去中心化的场景。另外在网络节点数少的场景，选举的见证人的代表性也不强。

（4）分布式一致性算法

分布式一致性算法是基于传统的分布式一致性技术。其中有分为解决拜占庭将军问题的拜占庭容错算法，如 PBFT。另外解决非拜占庭问题的分布式一致性算法（Pasox、Raft），详细见本书第 5 章的共识算法。该类算法目前是联盟链和私有链链场景中常用的共识机制。

- 优点：实现秒级的快速共识机制，保证一致性。
- 缺点：去中心化程度不如公有链上的共识机制；更适合多方参与的多中心商业模式。

3. 解锁脚本

脚本是区块链上实现自动验证、自动执行合约的重要技术。每一笔交易的每一项输出严格意义上并不是指向一个地址，而是指向一个脚本。脚本类似一套规则，它约束着接收方怎样才能花掉这个输出上锁定的资产。

交易的合法性验证也依赖于脚本。目前它依赖于两类脚本：锁定脚本与解锁脚本。锁定脚本是在输出交易上加上的条件，通过一段脚本语言来实现，位于交易的输出。解锁脚本与锁定脚本相对应，只有满足锁定脚本要求的条件，才能花掉这个脚本上对应的资产，位于交易的输入。通过脚本语言可以表达很多灵活的条件。解释脚本是通过类似我们编程领域里的“虚拟机”，它分布式运行在区块链网络里的每一个节点。

比特币的脚本目前常用的主要分为两种，一种是普通的 P2PKH（Pay-to-Public-Key-Hash），即支付给公钥的哈希地址，接收方只需要使用地址对应的私钥对该输出进行签名，即可花掉该输出。另一种是 P2SH（Pay-to-Script-Hash），即支付脚本的哈希。以多重签名来举例，它要求该输出要有 N 把私钥中的 M 把私钥（$M \leq N$）同时签名才能花掉该资产，它类似于现实生活中需要多把钥匙才能同时打开的保险柜，或是多人签名才能使条约生效一样，只是它是自动执行。

比如在比特币中，P2PKH 的脚本规则如下：

```
Pubkey script: OP_DUP OP_HASH160 <PubKeyHash> OP_EQUALVERIFY OP_CHECKSIG
Signature script: <sig> <pubkey>
```

P2SH 的脚本规则如下：

```
Pubkey script: OP_HASH160 <Hash160(redeemScript)> OP_EQUAL
Signature script: <sig> [sig] [sig...] <redeemScript>
```

在上述的两种脚本规则里，Pubkey script 代表锁定脚本，Signature script 代表解锁脚本。OP_ 开头的单词是相关的脚本命令，也是“虚拟机”所能解析的指令。这些命令规则根据 Pubkey script 的不同来进行划分，它也决定解锁脚本的规则。

比特币中的脚本机制相对简单，只是一个基于堆栈式的、解释相关OP指令的引擎，能够解析的脚本规则并不是太多，不能实现很复杂的逻辑。但它为区块链可编程提供了一个原型，后续一些可编程区块链项目其实是基于脚本的原理发展起来的，比如以太坊就是深入增强了脚本机制，脚本机制里不再单单是简单的OP指令，而是支持脚本的一套图灵完备语言，该脚本语言可以通过“虚拟机”去执行。以太坊实现了一个支持图灵完备脚本语言的区块链平台。

脚本的机制对于区块链来说非常重要，它类似于区块链技术提供的一个扩展接口，任何人都可以基于这个接口开发基于区块链技术的应用，比如智能合约的功能。脚本机制也让区块链技术作为一项底层协议成为可能。未来很多基于区块链的颠覆性应用，都有可能通过区块链的脚本语言来完成。

4. 交易规则

区块链的交易就是构成区块的基本单位，也是区块链负责记录的实际有效内容。一个区块链交易可以是一次转账，也可以是智能合约的部署等其他事务。

就比特币而言，交易即指一次支付转账。其交易规则如下：

1）交易的输入和输出不能为空。

2）对交易的每个输入，如果其对应的UTXO输出能在当前交易池中找到，则拒绝该交易。因为当前交易池是未被记录在区块链中的交易，而交易的每个输入，应该来自确认的UTXO。如果在当前交易池中找到，那就是双花交易。

3）交易中的每个输入，其对应的输出必须是UTXO。

4）每个输入的解锁脚本（unlocking script）必须和相应输出的锁定脚本（locking script）共同验证交易的合规性。

对于以太坊来说，交易还可能是智能合约的部署。交易规则就确定了符合一定语法规则的合约才能被部署在区块链上。

5. 交易优先级

区块链交易的优先级由区块链协议规则决定。对于比特币而言，交易被区块包含的优先次序由交易广播到网络上的时间和交易额的大小决定。随着交易广播到网络上的时间的增长，交易的链龄增加，交易的优先级就被提高，最终会被区块包含。对于以太坊而言，交易的优先级还与交易的发布者愿意支付的交易费用有关，发布者愿意支付的交易费用越高，交易被包含进区块的优先级就越高。

6. Merkle 证明

Merkle证明的原始应用是比特币系统（Bitcoin），它是由中本聪（Satoshi Nakamoto）

在 2009 年描述并且创造的。比特币区块链使用了 Merkle 证明，为的是将交易存储在每一个区块中。使得交易不能被篡改，同时也容易验证交易是否包含在一个特定区块中，Merkle 树说明详见 4.2 节。比特币的 Merkle 证明树如图 2-11 所示。

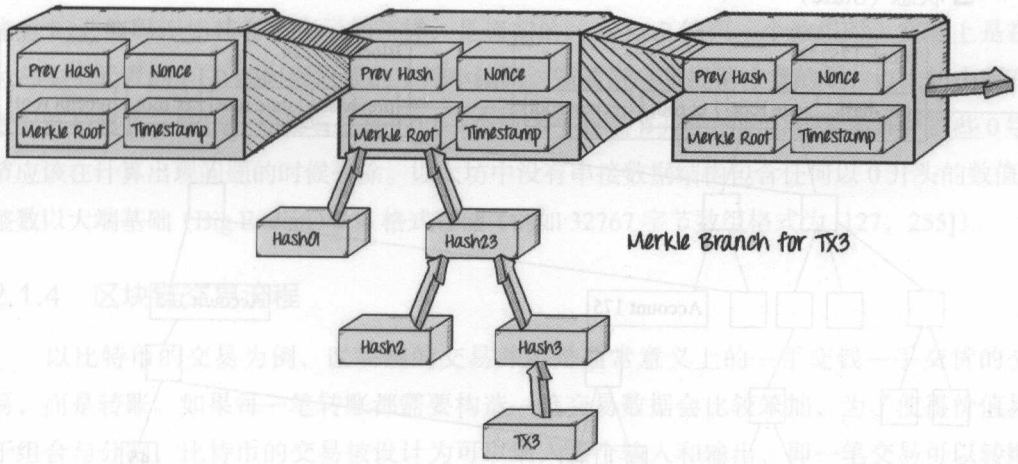


图 2-11　比特币的 Merkle 证明树

Merkle 树的一个重要使用场景就是快速支付验证，也就是中本聪描述的“简化支付验证”（SPV）的概念：轻量级节点（light client）不用下载每一笔交易以及每一个区块，可以仅下载链的区块头，每个区块中仅包含下述 5 项内容，数据块大小为 80 字节。

- 上一区块头的哈希值
- 时间戳
- 挖矿难度值
- 工作量证明随机数（nonce）
- 包含该区块交易的 Merkle 树的根哈希

如果一个轻客户端希望确定一笔交易的状态，它可以简单地要求一个 Merkle 证明，显示出一个在 Merkle 树特定的交易，其根是在主链（main chain，非分叉链）上的区块头。

Merkle 证明可以让区块链得到更广阔的应用，但比特币的轻客户有其局限性。虽然可以证明包含的交易，但无法证明任何当前的状态（例如：数字资产的持有，名称注册，金融合约的状态等）。一笔交易影响的确切性质（precise nature）可以取决于此前的几笔交易，而这些交易本身则依赖于更为前面的交易，所以最终你需要验证整个链上的每一笔交易。为了解决这个问题，以太坊进行了更进一步的创新。

以太坊的每一个区块头中并非只包含一棵 Merkle 树，而是包含了 3 棵 Merkle 树

（见图 2-12），分别对应了以下 3 种对象：

- 交易（Transactions）
- 收据（Receipts，基本上，它是展示每一笔交易影响的数据条）
- 状态（State）

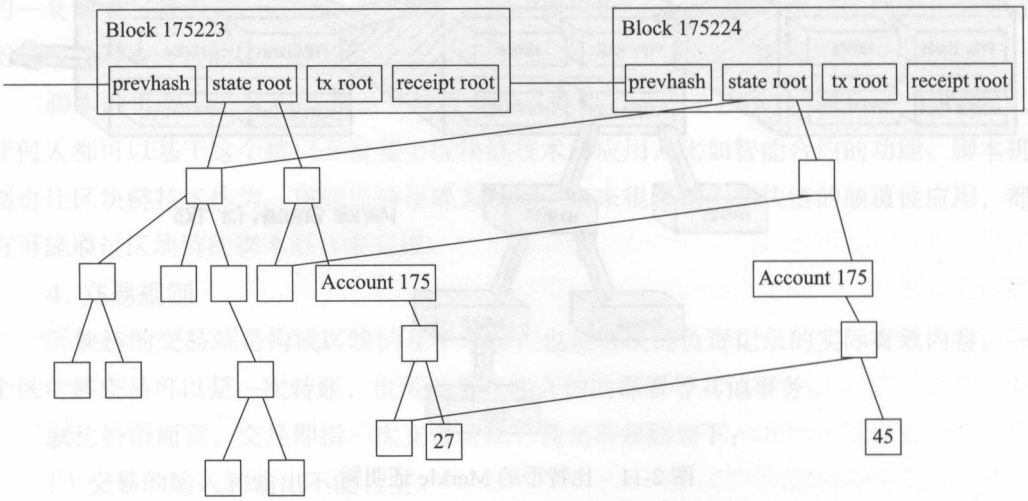


图 2-12　以太坊的 Merkle 证明树

这三棵树允许轻客户端轻松地进行并核实以下类型的查询答案：

1）这笔交易被包含在特定的区块中了吗？

2）告诉我这个地址在过去 30 天中，发出 X 类型事件的所有实例（例如，一个众筹合约完成了它的目标）。

3）目前我的账户余额是多少？

4）这个账户是否存在？

5）假装在这个合约中运行这笔交易，它的输出会是什么？

第一种是由交易树（transaction tree）来处理的；第 3 和第 4 种则是由状态树（state tree）负责处理，第 2 种则由收据树（receipt tree）处理。计算前 4 个查询任务是相当简单的。在服务器简单地找到对象，获取梅克尔分支，并通过分支来回复轻客户端。第 5 种查询任务同样也是由状态树处理。

7. RLP

RLP（Recursive Length Prefix，递归长度前缀编码）是 Ethereum 中对象序列化的一个主要编码方式，其目的是对任意嵌套的二进制数据的序列进行编码。

以太坊中的所有数据都以“递归长度前缀编码”（Recursive Length Prefix encoding，

RLP）形式存储，这种编码格式将任意长度和维度的字符串构成的数组串连接成字符串。例如，['dog', 'cat'] 被串接（以字节数组格式）为 [130，67，100，111，103，67，99，97，116]；其基本的思想是把数据类型和长度编码成一个单独的字节放在实际数据的前面（例如‘dog’的字节数组编码为 [100，111，103]，于是串接后就成了 [67，100，111，103]）。注意 RLP 编码正如其名字表示的一样，是递归的；当 RLP 编码一个数组时，实际上是在对每一个元素的 RLP 编码级联成的字符串编码。需要进一步提请注意的是，以太坊中所有数据都是整数；所以，如果有任何的以一个或多个 0 字节开头的哈希或者地址，这些 0 字节应该在计算出现问题的时候去除。以太坊中没有串接数据结构包含任何以 0 开头的数值。整数以大端基础（Big Endian）256 格式存储（例如 32767 字节数组格式为 [127，255]）。

2.1.4　区块链交易流程

以比特币的交易为例，区块链的交易并不是通常意义上的一手交钱一手交货的交易，而是转账。如果每一笔转账都需要构造一笔交易数据会比较笨拙，为了使得价值易于组合与分割，比特币的交易被设计为可以纳入多个输入和输出，即一笔交易可以转账给多个人。从生成到在网络中传播，再到通过工作量证明、整个网络节点验证，最终记录到区块链，就是区块链交易的整个生命周期。整个区块链交易流程如图 2-13 所示。

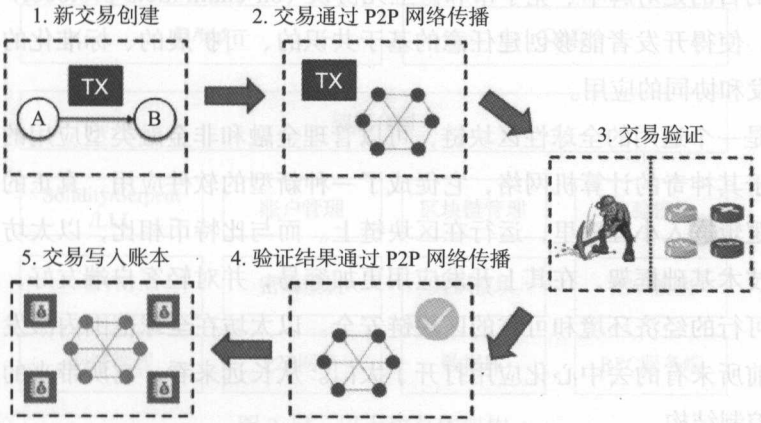


图 2-13　区块链交易流程

- 交易的生成。所有者 A 利用他的私钥对前一次交易和下一位所有者 B 签署一个数字签名，并将这个签名附加在这枚货币的末尾，制作成交易单。
- 交易的传播。A 将交易单广播至全网，每个节点都将收到的交易信息纳入一个区块中。

- 工作量证明。每个节点通过相当于解一道数学题的工作量证明机制，从而获得创建新区块的权力，并争取得到数字货币的奖励。
- 整个网络节点验证。当一个节点找到解时，它就向全网广播该区块记录的所有盖时间戳交易，并由全网其他节点核对。
- 记录到区块链。全网其他节点核对该区块记账的正确性，没有错误后他们将在该合法区块之后竞争下一个区块，这样就形成了一个合法记账的区块链。

2.2 以太坊

2.2.1 什么是以太坊

自 2008 年比特币出现以来，数字货币的存在已经渐渐为一部分人所接受。人们也积极展开了基于比特币的商业应用的思考与开发。但是随着应用的扩展，人们发现比特币的设计只适合虚拟货币场景，由于存在着非图灵完备性、缺少保存状态的账户概念，以及 PoW 挖矿机制所带来的资源浪费和效率问题，在很多区块链应用场景下并不适用。人们需要一个新的基于区块链的具有图灵完备性、高效共识机制、支持更多应用场景的智能合约开发平台。以太坊在这种情况下应运而生。

以太坊的目的是对脚本、竞争币和链上元协议（on-chain meta-protocol）等概念进行整合和提高，使得开发者能够创建任意的基于共识的、可扩展的、标准化的、图灵完备的、易于开发和协同的应用。

以太坊是一个通用的全球性区块链，可以管理金融和非金融类型应用的状态。以太坊的新颖在于其神奇的计算机网络，它促成了一种新型的软件应用，真正的去中心化应用。将信任逻辑嵌入小程序里，运行在区块链上。而与比特币相比，以太坊建立了一种新的密码学技术基础框架，在其上开发应用更加容易，并对轻客户端友好，同时允许应用共享一个可行的经济环境和可靠的区块链安全。以太坊在全球范围内激发了商业和社会创新，为前所未有的去中心化应用打开了大门。从长远来看，它所带来的改变将影响全球经济和控制结构。

以太坊是个平台和编程语言，包括数字货币以太币（Ether），以及用来构建和发布分布式应用的以太脚本（EtherScript）。

以太币和著名的数字货币比特币有非常多的相似之处。两者均为数字货币且无法伪造，都以去中心化的方式运行来保证货币供应不被某一方所控制。以太坊的另一半重要特性是提供一个完整的编程语言环境，有时也被叫作以太脚本。我们都知道，编程语言

是人类用来控制计算机工作的。因此，用任何编程语言写好的指令对计算机来说都是准确无误没有歧义的。也就是说，计算机如何执行一段代码是没有二义性的。在同样的条件下，一段代码总是会按照既定的步骤执行。这种特性正是人类现行法律与合约所缺失的。因此，有了以太脚本之后，我们就可以制定没有二义性的合约了。

从最底层角度来看，以太坊是一个多层的、基于密码学的开源技术协议。它的不同功能模块通过设计进行了全面的整合，作为一个整体，它是一个创建和部署去中心化应用的综合平台。虽然，以太坊看起来像由多个互相联系的开源项目构成的混合体，但是它的进化一直被明确目标所引导，所以各个组件可以协同地组装在一起。

同时，以太坊也是区块链与智能合约的完美结合，是智能合约的完整解决方案，被设计成了一个通用的去中心化平台，拥有一套完整的、可以扩展其功能的工具，在 P2P 网络、加密、HttpClient 等技术的支持下实现了一个类似于比特币的区块链。它通过工作量证明机制实现共识，由矿工挖矿，通过对新的网络协议的制定实现对区块链的同步等操作。不同于比特币的是，在以太坊上可以任意编写智能合约，通过智能合约实现强大的功能，实现去中心化应用的开发。在以太坊上部署的智能合约运行在以太坊特有的虚拟机上，通过以太坊虚拟机和 RPC 接口与底层区块链进行交互。

以太坊的总体系统架构如图 2-14 所示。

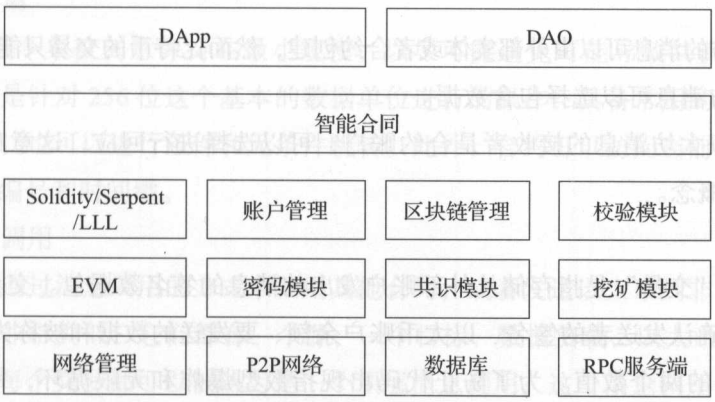


图 2-14　以太坊总体架构

2.2.2　以太坊技术

1. 以太坊核心概念

（1）以太坊虚拟机

以太坊虚拟机（EVM）是以太坊中智能合约的运行环境。它是以太坊项目中的另一

个主要创新。有人说 EVM “位于区块链之上”，实际上它是由许多互相连接的计算机组成的。任何人都可以上传程序，并让这些程序自动执行，同时保证现在和所有以前的每个程序的状态总是公共可见的。这些程序运行在区块链上，严格地按照 EVM 定义的方式继续执行。所以任何人都可以为所有权、交易格式和状态转换函数创建商业逻辑。

（2）账户

以太坊中有两类账户，它们共用同一个地址空间。外部账户，该类账户被公钥 - 私钥对控制。合约账户，该类账户被存储在账户中的代码控制。外部账户的地址是由公钥决定的，合约账户的地址是在创建合约时由合约创建者的地址和该地址发出过的交易数量计算得到。两类账户的唯一区别是：外部账户没有代码，人们可以通过创建和签名一笔交易从一个外部账户发送消息。每当合约账户收到一条消息，合约内部的代码就会被激活，允许它对内部存储进行读取、写入、发送其他消息和创建合约。

以太坊的账户包含 4 个部分：①随机数，用于确定每笔交易只能被处理一次的计数器；②账户目前的以太币余额；③账户的合约代码（如果有的话）；④账户的存储（默认为空）。

（3）消息

以太坊的消息在某种程度上类似于比特币的交易，但是两者之间存在 3 点重要的不同。

1）以太坊的消息可以由外部实体或者合约创建，然而比特币的交易只能从外部创建。

2）以太坊消息可以选择包含数据。

3）如果以太坊消息的接收者是合约账户，可以选择进行回应，这意味着以太坊消息也包含函数概念。

（4）交易

以太坊中“交易”是指存储从外部账户发出的消息的签名数据包。交易包含消息的接收者、用于确认发送者的签名、以太币账户余额、要发送的数据和被称为 STARTGAS 和 GASPRICE 的两个数值。为了防止代码出现指数型爆炸和无限循环，每笔交易需要对执行代码所引发的计算步骤做出限制。STARTGAS 就是通过需要支付的燃料来对计算步骤进行限制，GASPRICE 是每一计算步骤需要支付矿工的燃料的价格。

（5）Gas

以太坊上的每笔交易都会被收取一定数量的燃料 Gas，设置 Gas 的目的是限制交易执行所需的工作量，同时为交易的执行支付费用。当 EVM 执行交易时，Gas 将按照特定规则被逐渐消耗。Gas 价格由交易创建者设置，发送账户需要预付的交易费用 =

GASPRICE * Gas amount。如果执行结束还有 Gas 剩余，这些 Gas 将被返还给发送账户。无论执行到什么位置，一旦 Gas 被耗尽就会触发一个 out-of-gas 异常。同时，当前调用帧所做的所有状态修改都将被回滚。

（6）存储、主存和栈

每个账户都有一块永久的内存区域，被称为存储，其形式为 key-value，key 和 value 的长度均为 256 位。在合约里，不能遍历账户的存储。相对于主存和栈，存储的读操作开销较大，修改存储甚至更多。一个合约只能对它自己的存储进行读写。

第二个内存区被称为主存。合约执行每次消息调用时都有一块新的被清除过的主存。主存可以按字节寻址，但是读写的最小单位为 32 字节。操作主存的开销随着主存的增长而变大。

EVM 不是基于寄存器的，而是基于栈的虚拟机。因此所有的计算都在一个称为栈的区域内执行。栈最大有 1024 个元素，每个元素有 256 位。对栈的访问只限于其顶端，允许复制最顶端的 16 个元素中的一个到栈顶，或者是交换栈顶元素和下面 16 个元素中的一个。所有其他操作都只能取最顶的一个或几个元素，并把结果压在栈顶。当然可以把栈里的元素放到存储或者主存中。但是无法只访问栈里指定深度的那个元素，在那之前必须把指定深度之上的所有元素都从栈中移除才行。

（7）指令集

EVM 的指令集被刻意保持在最小规模，以尽可能避免可能导致共识问题的错误。所有的指令都是针对 256 位这个基本的数据单位进行的操作，具备常用的算术、位、逻辑和比较操作，也可以进行条件和无条件跳转。此外，合约可以访问当前区块的相关属性，比如它的编号和时间戳。

（8）消息调用

合约可以通过消息调用的方式来调用其他合约，或者发送以太币到非合约账户。消息调用和交易非常类似，它们都有一个源，一个目标，数据负载，以太币，Gas 和返回数据。事实上每个交易都可以被认为是一个顶层消息调用，这个消息调用会依次产生更多的消息调用。

一个合约可以决定剩余 Gas 的分配。比如内部消息调用时使用多少 Gas，或者期望保留多少 Gas。如果在内部消息调用时发生了 out-of-gas 异常或者其他异常，合约将会得到通知，一个错误码被压入栈中。这种情况只是内部消息调用的 Gas 耗尽。在 solidity 中，这种情况下发起调用的合约默认会触发一个人工异常，这个异常会打印出调用栈。

就像之前说过的，被调用的合约（发起调用的合约也一样）会拥有崭新的主存，并能够访问调用的负载。调用负载被存储在一个单独的被称为 calldata 的区域。调用执行结束后，返回数据将被存放在调用方预先分配好的一块内存中。调用层数被限制为 1024。因此对于更加复杂的操作，我们应该使用循环而不是递归。

（9）代码调用和库

以太坊中存在一种特殊类型的消息调用，被称为 callcode。它跟消息调用几乎完全一样，只是加载来自目标地址的代码将在发起调用的合约上下文中运行。这意味着一个合约可以在运行时从另外一个地址动态加载代码。存储，当前地址和余额都指向发起调用的合约，只有代码是从被调用地址获取的。这使得 Solidity 可以实现“库”。可复用的库代码可以应用在一个合约的存储上，可以用来实现复杂的数据结构，从而使智能合约更加的强大。

2. 以太坊的状态转换

以太坊的状态转换是指在一个交易（TX）发生时，以太坊从一个正确状态（S）转变到下一个正确状态（S’）的转换过程，如图 2-15 所示。对于交易而言，为了防止代码的指数型爆炸和无限循环，每笔交易需要对执行代码所引发的计算步骤做出限制。STARTGAS 就是限制，GASPRICE 是每一计算步骤需要支付矿工的费用价格。

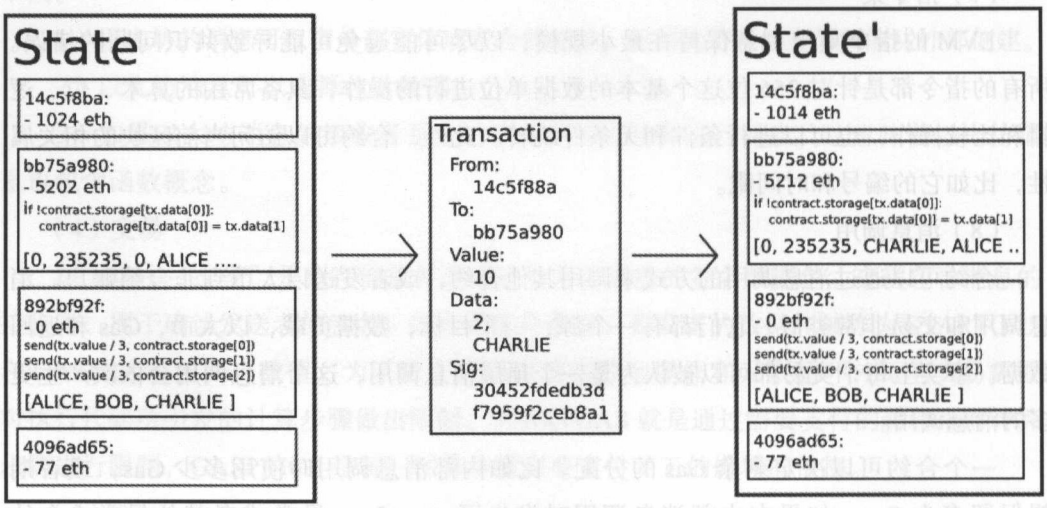


图 2-15 以太坊状态转换

以太坊的状态转换函数为 APPLY(S, TX) -> S’，可以定义如下：

1）检查交易的格式是否正确，签名是否有效，以及随机数是否与发送者账户的随

机数匹配。如否，返回错误。

2）计算交易费用 fee = STARTGAS * GASPRICE，并从签名中确定发送者的地址。从发送者的账户中减去交易费用和增加发送者的随机数。如果账户余额不足，返回错误。

3）设定初值 Gas = STARTGAS，并根据交易中的字节数减去一定量的燃料值。

4）从发送者的账户转移价值到接收者账户。如果接收账户还不存在，创建此账户。如果接收账户是一个合约，运行合约的代码，直到代码运行结束或者燃料用完。

5）如果因为发送者账户没有足够的费用或者代码执行耗尽燃料导致价值转移失败，恢复原来的状态，但是还需要支付交易费用，交易费用加至矿工账户。

6）若代码执行成功，将所有剩余的燃料归还给发送者，消耗掉的燃料作为交易费用发送给矿工。

例如，假设一个合约的代码如下：

```
if !contract.storage[msg.data[0]]:
contract.storage[msg.data[0]] = msg.data[1]
```

需要注意的是，在现实中合约代码是用底层以太坊虚拟机（EVM）代码写成的。上面的合约是用我们的高级语言 Serpent 语言写成的，它可以被编译成 EVM 代码。假设合约存储器开始时是空的，一个值为 10 以太、燃料为 2000、燃料价格为 0.001 以太并且两个数据字段值为 [2，'CHARLIE'] 的交易发送后，状态转换函数的处理过程如下：

1）检查交易是否有效，格式是否正确。

2）检查交易发送者是否至少有 2000 × 0.001=2 个以太币。如果有，从发送者账户中减去 2 个以太币。

3）初始设定 Gas=2000，假设交易长为 170 字节，每字节的费用是 5，减去 850，所以还剩 1150。

4）从发送者账户减去 10 个以太币，为合约账户增加 10 个以太币。

5）运行代码。在这个合约中，运行代码很简单：它检查合约存储器索引为 2 处是否已使用，注意到它未被使用，然后将其值置为 CHARLIE。假设这消耗了 187 单位的燃料，于是剩余的燃料为 1150–187 = 963。

6）向发送者的账户增加 963 × 0.001=0.963 个以太币，返回最终状态。

如果没有合约接收交易，那么所有的交易费用就等于 GASPRICE × 交易的字节长度，交易的数据就与交易费用无关了。另外，需要注意的是，合约发起的消息可以对它们产生的计算分配燃料限额，如果子计算的燃料用完了，它只恢复到消息发出时的状

态。因此，就像交易一样，合约也可以通过对它产生的子计算设置严格的限制，保护它们的计算资源。

3. 以太坊客户端

为了测试各种语言对以太坊的支持，同时使更多的人能够参与以太坊的开发及使用，目前有 4 种语言编写的以太坊的客户端。它们分别是用 Go 语言实现的客户端 Geth，用 C++ 实现的客户端 Eth，用 Python 语言实现的客户端 Pyethapp 和用 Java 实现的客户端 EthereumJ。其中，Go 语言版是以太坊官方一直维护并推荐使用的客户端。

以太坊包括一个专用的客户端浏览器，使得用户可以运行各种各样的去中心化应用（DApp），发布智能合约。这一浏览器被称为 Mist，它易于使用，降低了用户使用门槛，从而使得 DApp 和智能合约能够被大量用户使用。它的作用等同于浏览器之于互联网或者 iTunes 之于数字化内容下载。Mist 由特殊的安全层、密钥管理、去中心化账户管理和与区块链相关的组件几部分组成。这一切使得 Mist 成为普通用户运行或者管理区块链去中心化应用不可或缺的工具，普通用户不需要理解技术方面的东西。

从用户体验角度而言，可以在 Mist 中使用 DApp，就像通过常规浏览器与网站进行交互一样。例如，一个纯 DApp（例如预测市场 Augur）就可以在以太坊 Mist 浏览器中运行。当然，这些服务也可以通过一个常规浏览器以更加传统的 Web 2.0 的方式实现。

2.2.3 以太坊智能合约

1. 智能合约

智能合约是由尼克萨博提出的理念，几乎与互联网同龄。但是由于缺少可信的执行环境，智能合约并没有被应用到实际产业中。自从比特币诞生后，人们认识到比特币的底层技术区块链天生可以为智能合约提供可信的执行环境，以太坊首先实现了区块链和智能合约的完整契合。

以太坊是内置有图灵完备[⊖]编程语言的区块链，通过建立抽象的基础层，使得任何人都能够创建合约和去中心化应用，并在其中设立他们自由定义的所有权规则、交易方式和状态转换函数。建立一个代币的主体框架只需要两行代码就可以实现，诸如货币和信誉系统等其他协议只需要不到 20 行代码就可以实现。智能合约就像能在以太坊的平台上创建的包含价值而且只有满足某些条件才能打开的加密箱子，并且因为图灵完备性、价值意识（value-awareness）、区块链意识（blockchain-awareness）和记录多状态所增加的功能而比比特币脚本所能提供的智能合约强大得多。

⊖ 关于图灵完备的相关内容请参见 3.3 节。

2. 开发语言

以太坊的软件开发语言是其最大特性之一，因为对区块链进行编程是一项首要目标。以太坊具有 4 种专用语言：Serpent（受 Python 启发）、Solidity（受 JavaScript 启发）、Mutan（受 Go 启发）和 LLL（受 Lisp 启发），都是为面向合约编程而从底层开始设计的语言。

作为以太坊的高级编程语言，Serpent 的设计非常类似于 Python。它被设计为最大可能地简洁和简单，将低级语言的高效优势与编程风格中的易用性相结合。

Solidity 是以太坊的首选语言，它内置了 Serpent 的所有特性，但是语法类似于 JavaScript。Solidity 充分利用了现有数以百万程序员已掌握 JavaScript 这一现状，降低了学习门槛，易于被掌握和使用。

以太坊区块链的另一关键特征是它的“图灵完备性”，这保证了以太坊可以解决所有的计算问题。更加准确地说，它是“半”图灵完备的，因为它是通过对计算量设置上限，它避免了完全图灵完备语言存在的无法停机问题。此外，因为以太坊的语言是为区块链专门设计的，它有账户的概念，使得它在交易的可视化和查询账户状态方面提供了实时性。这是一个受人欢迎的功能，但对比特币而言实现起来具有一定的挑战。在比特币上，由于只有 UTXO 而没有账户的概念，我们需要导入区块链数据库，解析所有的交易，并为了抽取出在区块链上的某个用户的交易情况而查询交易。而用以太坊，我们则可以在实时的区块链上，根据一个地址情况实时查看当前账户情况和交易状态。

3. 代码执行

以太坊合约的代码是使用低级的基于堆栈的字节码的语言写成的，被称为“以太坊虚拟机代码”或者“EVM 代码”。代码由一系列字节构成，每一个字节代表一种操作。一般而言，代码执行是无限循环，程序计数器每增加一（初始值为零）就执行一次操作，直到代码执行完毕或者遇到错误、STOP 或者 RETURN 指令。操作可以访问 3 种存储数据的空间：

1）堆栈，一种后进先出的数据存储，入栈、出栈的基本单位为 32 字节。

2）内存，可无限扩展的字节队列。

3）合约的长期存储，一个密钥 / 数值的存储，其中密钥和数值都是 32 字节大小。与计算结束即重置的堆栈和内存不同，存储内容将长期保持。

代码可以像访问区块头数据一样访问数值、发送和接收到的消息中的数据，代码还可以返回数据的字节队列作为输出。EVM 代码的正式执行模型非常简单。当以太坊虚拟机运行时，它的完整的计算状态可以由元组（block_state，transaction，message，

code，memory，stack，pc，gas）来定义，这里 block_state 是包含所有账户余额和存储的全局状态。每轮执行时，通过调出代码的第 pc（程序计数器）个字节，每个指令如何影响元组都有定义。例如，ADD 将两个元素出栈并将它们的和入栈，将 Gas 减 1 并将 pc 加 1；stack 将顶部的两个元素出栈，并将第 2 个元素插入由第 1 个元素定义的合约存储位置，同样减少最多 200 的 Gas 值，并将 pc 加 1。虽然有许多方法通过即时编译去优化以太坊，但以太坊的基础性的实施可以用几百行代码实现。

2.2.4 以太坊的去中心化应用

1. 什么是 DApp

一个 DApp 是由智能合约和客户端代码构成的。智能合约就像加密的包含价值的箱子。只有当特定条件被满足时它才被打开，它封装了一些逻辑、规则、处理步骤或者双方间的协议。

从架构角度而言，DApp 非常类似于传统的 Web 应用。主要区别是：在传统 Web 应用中，客户端有 JavaScript 代码，由用户在自己的浏览器中执行；服务器端的代码由主机运行。但是在一个 DApp 中，它的智能逻辑运行在区块链上，客户端代码运行在特殊浏览器 Mist 里面。

2. 应用举例

正如苹果电脑公司的 App Store 给众多公司和个人提供了极佳的商业机会一样，以太坊这样的平台上一定会出现类似“愤怒的小鸟”这样的成功范例，对于 IT 导向的创业公司和个人，这是一个广阔的空间。下面举几个例子，看看这个空间中正在孕育的项目。

Augur（www.augur.net），一个正在开发去中心化预测系统。Augur 在英文中的意思是“预言家”，用户可以在这个应用上对各种事件打赌并下注，例如希拉里会不会赢得 2016 年美国的大选；2016 年中国的 GDP 增长会不会超过 7%；上证指数 2020 年之前会不会超 10000 点，等等。对于参与者而言，如果预测正确，则将获得经济上的回报；而对于社会整体而言，Augur 便成了一个群体智慧的收集器，在它上面的下注信息反映了人们对于未来某事件发生可能性的最佳评估。当你打开搜索引擎，输入“×××会不会赢得 2020 年美国的大选”的时候，可能搜到这样的结果：“Augur：该事件发生的可能性为 46.6%”。

Maker（www.makerdao.com），一个正在开发中的金融类去中心化自治组织。Maker 是一个去中心化自治组织（Decentralized Autonomous Organization，DAO），它维护一系列用于金融服务的合约软件（即智能合约），其中的一个软件是贷券信贷系统（The Dai Credit System）。贷券信贷系统是一个通过抵押进行借贷的应用，当用户在区块链上拥有

众多资产时，一个必然的需求就是，在不出售资产的情况下通过抵押借款获得资金，类似于房屋抵押贷款。

WeiFund，一个正在开发中的去中心化众筹平台。这个众筹平台的好处是资金不需要第三方托管，而是由程序托管，因此能够确保资金 100% 安全。如果在限定时间内众筹项目的资助超过预定目标，则程序将资金发送给创业者；如果目标未达到，则将资金退回给众筹参与者。

Boardroom，一个正在开发中的 DAO 管理平台。去中心化自治组织与传统公司一样，也可以有股份的概念，通常就是 DAO 的代币，这些代币又可以有投票权。与上一个例子中的 WeiFund 结合，可以发生有趣的事：如果众筹募资的目标达到，则 WeiFund 可以将资金发送给 Boardroom，并将该众筹项目的参与者组成 DAO。如此一来，这些参与者便可以通过投票来控制资金的具体使用，而不是简单地将资金一次性发放给创业者，这样便大大避免了来自创业者的道德风险。

Ujo Music，一个音乐版权管理平台，测试版。Ujo Music 作为音乐版权管理平台，可以直接在歌曲的创作者与消费者之间建立直接的联系，从而省去了中间商的费用提成。更进一步，歌手甚至可以将自己的作品在 WeiFund 上进行 IPO 上市，如此一来，持有歌手代币的投资者就会分得歌手今后歌曲的销售收入，他们是歌手天然的铁杆粉丝团。帮助歌手推广音乐也不仅仅出于经济动机，同时也是出于情感上的支持，试想在周杰伦默默无闻的时候就持有他的代币，该是一件多么令粉丝感到骄傲的事。同时代币也是歌手与粉丝团之间互动的媒介，例如歌手可以让持有代币的粉丝优先购买演唱会的门票，如此，粉丝经济会更加红火。

贷券（Decentralized Autonomous Insured Bond）简称 Dai 或者 Dai Bond，是一种可转让的、彼此等价可互换的“加密债券”，它流通于信贷系统中，使用者无需事先认证，同时又是低风险的。贷券的发行人（借款方）将在以太坊区块链上的以太等数字加密资产作为抵押品来发行贷券，再将这些贷券在市场上卖给贷券持有人以换回流动性好的资产。贷券的持有人之所以买入贷券是为了赚取收益，或者是为了将贷券当作价值稳定的加密货币来流通使用。

2.3　基于区块链的电子货币

2.3.1　元币平台

元币（metacoin）单词前缀“meta-”意为“在其中”。所以元币是衍生于现有加密

货币体系之上，更专注于业务系统的代币种类。

（1）彩币

彩币（Colored Coins）是一种建立在比特币数据块上的高级衍生应用协议。彩币建立在 P2P 网络之上，具有去中心化的所有优势，可以用来进行任何虚拟财产和现实资产的交易，并且比传统金融交易更加快捷和方便，更重要的是，它没有手续费。彩币由一种特殊的钱包负责管理，进行记录和阐释附加在彩币中的元数据。通过这种钱包，用户可以通过添加标签，将一定量的比特币转化为彩币，使其具有特殊意义。而这种特殊意义，往往使其价值提升。相关元数据是由用户来定义的，并且使比特币转化为彩币。彩币一旦定义，便可以进行买卖、分销、积累或者分红。当然，也可以去颜色化，回归比特币。

（2）万事达币

万事达币（mastercoin）发布于 2013 年 6 月 24 日，是建立在比特币协议之上的二代币，旨在帮助用户创建并交易加密货币以及其他类型的智能合同。万事达币不仅仅是一种代币，更像是一个去中心化的财产交易、合同签署、用户货币、智能财产代币等的平台。打个形象的比方，就像 HTTP 运行在 TCP 层之上一样，万事达币是工作于比特币交易层之上的应用层协议。

（3）合约币

合约币（counterparty）是另一种建立在比特块链基础之上的协议层，它把比特币块链当成可信的时间戳服务和可信的信息发布证明。合约币在比特币中添加额外的信息记录，以此强化比特币的交易。合约币实现了用户货币、可交易代币、财经工具、去中心化交易等一些功能。

2.3.2 代币

尽管部分代币没有使用比特币，而是直接在区块链的基础上全新创建的，但事实上，大多数代币都是比特币衍生出来的。其中最值得一提的是莱特币（Litecoin），而莱特币本身又被作为代币的母币，衍生出更多新的代币。以同样方式诞生的还有代链这个概念。如同字面意思，两者的区别在于，代币用作货币目的，而代链用于非货币目的。本节介绍内容属于代币范畴。

代币的创建过程非常简单。2011 年 8 月，通过修改比特币的一些参数，加速货币的产生速度，世界上第一种代币 IXCoin 诞生。自此以后，代币发展极其迅速。到 2016 年，已经有超过 600 种代币发布，并且，大部分是莱特币的变种。它们与比特币的差异

主要存在于货币政策、工作量证明和共识机制以及所引入的新特性方面。其实，大多数代币之间差异甚微，甚至，有些代币的产生，只是发明者为了增加收入的一种手段，并不具备研究价值。但是，这些代币中还是有一些比较明显的例外，或者说有非常重要的创新值得我们学习。那么如何衡量一种代币呢？通常从如下几个方面思考：是否具有重大创新？是否能因为与比特币的区别从比特币吸引过来用户？是否针对市场或者应用？是否能吸引足够多的矿工来抵御联合作弊？财经和市场指标上，又可以从资本量、预计用户量、日交易量和流通性上来评价一种代币。

1. 通过货币参数修改而建立的代币

比特币有一些货币参数来预防货币通货膨胀。比特币限制为 2100 万主要货币单元的货币总量。许多代币修改了比特币的主要参数，从而实现了新的不同货币政策。最具代表性的有如下几种。

（1）莱特币

要介绍莱特币（Litecoin），就不得不提 Tenebrix，它是第一个应用不同工作量证明算法的加密货币。此算法被称作 scrypt，是专门为预防暴力破解密码而设计的。Tenebrix 并没有成功，却为莱特币的诞生奠定了基础。作为主要的成功代币，莱特币的核心是继承了 scrypt 算法，修改了块产生时间，使之从比特币的 10 分钟缩短到 2.5 分钟，从而成为一种轻型代币。拥护者们认为，相对于比特币，莱特币更适合零售业的交易。更重要的是，以莱特币为基础，后来延伸出来上百种类似的代币。

> 提示：块产生时间，2.5 分钟；总货币量，8400 万（2140 年）；共识算法，工作量证明 Scrypt；市值，96 961 万元（2016 年年中）。

（2）狗狗币

作为莱特币的变种，狗狗币（Dogecoin）发布于 2013 年 12 月。狗狗币的货币政策特点是发行迅速，货币配额巨大。

> 提示：块产生时间，1 分钟；总货币量，10 000 亿（2016 年年中）；共识算法，工作量证明 Scrypt；市值，8201 万元（2016 年年中）。

（3）运输币

运输币（Freicoin）发布于 2012 年 7 月，是一种滞期代币。其储存额具有负利率。运输币不鼓励持有，它收取 4.5% 的年费来促进消费。运输币之所以出名，是因为它所

采用的货币政策与比特币的通货紧缩政策完全相反。作为代币，运输币并没有取得成功，却成为了供代币借鉴的多样性货币政策的重要一员。

提示：块产生时间，10 分钟；总货币量，1 亿（2140 年）；共识算法，工作量证明 SHA256；市值，13 万美元（2014 年年中）。

2. 基于共识机制的创新而建立的代币

比特币的共识机制是建立在用 SHA256 算法加密的作业证据基础之上的。从此，共识机制的创新进入了一个狂热的发展阶段。一些代币采用了各式各样的算法，如：Scrypt、Scrypt-N、Skein，Groestl、SHA3、X11、Blake 等。还有的代币同时采用其中的几种算法。2013 年，作为工作量证明的替代品，权益证明（Proof of Stake）被提出，为许多新代币的诞生提供了基础。权益证明系统中，拥有者可以将货币像股票一样抵押生息。它在某种程度上类似于存款证明，持有者可以保留所持货币的一部分，同时，以利息支付和交易费的形式赚取投资回报。

（1）点点币

点点币（Peercoin）诞生于 2012 年 8 月，是第一种综合使用工作量证明和权益证明机制来发行新币的代币。

提示：块产生时间，10 分钟；总货币量，不限；共识算法，工作量证明机制与权益证明结合；市值，5825 万元（2016 年年中）。

（2）Myriad

Myriad 发行于 2014 年 2 月，具有复合算法的特性，是第一个同时包含 5 种算法的加密货币。与比特币只能使用 SHA256 矿机挖矿不一样，Myriad 兼容 SHA256、Scrypt 等算法，同时兼容 GPU 及 CPU 挖矿。Myriad 对共识攻击具有更强的抵抗力，这是因为只有多种挖矿算法同时被攻击时才会受到攻击。

提示：块产生时间，平均 0.5 分钟；总货币量，20 亿（2024 年）；共识算法，工作量证明机制与多算法共用；市值，12 万美元（2014 年年中）。

（3）黑币

发行于 2014 年 2 月的黑币（Blackcoin）使用权益证明作为共识算法。黑币走进人

们的视野是因为它引入了多池这种可以根据利润差异在不同代币间自动切换的挖矿池。

提示：块产生时间，1 分钟；总货币量，不限；共识算法，权益证明；市值，1125 万元（2016 年年中）。

（4）维理币

维理币（VeriCoin）发行于 2014 年 5 月。它使用权益证明共识机制，并且采用根据市场供求动态调节的利率。同时，维理币也是第一种可以自动兑换成比特币的代币。

提示：块产生时间，1 分钟；总货币量，不限；共识算法，权益证明；市值，390 万元（2016 年年中）。

（5）未币

未币（NXT）彻底抛弃工作量证明，而仅仅使用权益证明共识机制，故 NXT 是一种全新的加密货币，因此不应称之为比特币的变种或者代币。NXT 具有许多先进特性，如：名称注册、去中心化的财产交易、整合分散的安全消息、股票代表等。因此，NXT 常被称作下一代数字加密货币。

提示：块产生时间，1 分钟；总货币量，不限；共识算法，权益证明；市值，3816 万元（2016 年年中）。

3. 通过双目标挖矿机制创新而建立的代币

比特币的工作量证明算法仅仅具有一个目的，那就是确保比特链的安全。尽管相对于传统支付系统的安全，这点挖矿开销并不高，但是被指是一种浪费。双目标工作量证明算法在产生工作量证明来满足安全的同时，解决一个具体的问题。但这样做也带来一定的风险。在向货币安全添加额外功能的同时，也影响了其供求曲线。

（1）质数币

质数币（Primecoin）发行于 2013 年 7 月，据称其“具有数学价值”。这种币尝试把虚拟货币中无目的的算法所浪费的资源利用起来。质数币的工作量证明机制有一定的科学价值。所以从某种意义上讲，质数币的矿工在挖矿的同时也促进了科学的进步。质数币是没有总量上限的。但是，质数币的产生速度很慢，这与大质数的计算困难特性有关。

提示：块产生时间，1 分钟；总货币量，不限；共识算法，工作量证明和质数链发现；市值，533 万元（2016 年年中）。

（2）治疗币

治疗币（Curecoin）发行于 2013 年 5 月。它将 SHA256 工作量证明算法与蛋白质折叠研究相结合，强调对生命科学和医学的贡献。其中蛋白质折叠是蛋白质生化反应的计算密集型模拟，目的在于发现新药。

提示：块产生时间，10 分钟；总货币量，不限；共识算法，工作量证明和蛋白质折叠研究；市值，194 万元（2016 年年中）。

（3）格雷德币

格雷德币（Gridcoin）发行于 2013 年 10 月。它支持基于 scrypt 的工作量证明算法。格雷德币通过伯克利的分布式计算项目（BOINC）进行包罗万象的科学探索，只有在挖矿的同时运行 BOINC 才能够获得较多的币。BOINC 是一种科研网格计算的开放协议，允许参与者贡献自己的空余计算力来进行一系列科研计算。所以，不同于质数币和治疗币那样目的单一，格雷德币可以通过 BOINC 这个计算平台帮助进行各种科学研究，包括寻找外星人、计算 RNA 结果等。

提示：块产生时间，2.5 分钟；总货币量，不限；共识算法，工作量证明和 BOINC 网格计算；市值，2858 万元（2016 年年中）。

4. 注重隐私性的代币

比特币常常被误认为具有隐私性。事实上，通过大数据分析，关联身份和比特币地址相对容易，从而通过连接彼此的地址，解析一个人的比特币消费习惯。所以，一些代币专注于强隐私性，试图直接解决此问题。

（1）零币

零币（Zerocoin）作为比特币之上保持隐私性的元币协议，是针对隐私性的第一个尝试。它诞生于 2013 年，仅是一种针对数字货币隐私性的理论方法。应用零币理论的代币叫作 Zerocash，至今仍在研究阶段，并未发行。

（2）CryptoNote

CryptoNote 公布于 2012 年 12 月，是一种为匿名数字货币提供基础的参考实践代

币。它的设计初衷就是为了不同的实践并且内置定时重置机制，这使得它本身不太可能成为一种货币。CryptoNote 的出名还来自于它不是比特币的翻版，而是全新从头设计实践的加密货币。基于 CryptoNote 衍生出来的数字货币还有 Bytecoin（BCN）、Aeon（AEON）、Boolberry（BBR）、DuckNote（DUCK）、Fantomcoin（FCN）、Monero（XMR）、MonetaVerde（MCN）和 Quazarcoin（QCN）等。更详细的 CryptoNote 衍生代币请见网页 https://en.wikipedia.org/wiki/File:Forks-tree-fixed.png。

（3）百特币

百特币（Bytecoin）发行于 2012 年 7 月，是第一个衍生自 CryptoNote 的代币（比 CryptoNote 本身的发行早）。百特币基于 CryptoNote 技术，提供一种可行的匿名货币。它从 CryptoNote 继承了环签名、非连接交易和抵制区块链分析的隐私性。之后，百特币还进行了包括多签名交易、安全更新等方面的改进。

提示：块产生时间，2 分钟；总货币量，1.84 亿元；共识算法，工作量证明 Cryptonight；市值，2909 万元（2016 年年中）。

（4）门罗币

门罗币（Monero，XMR）是 CryptoNote 的另外一种应用。相比于百特币，它具有略微平坦的发行曲线，在前 4 年就发行了 80%。它继承了 CryptoNote 所有的隐私性。

提示：块产生时间，1 分钟；总货币量，1844 万元；共识算法，工作量证明 Cryptonight；市值，2622 万元（2016 年年中）。

（5）黑暗币

黑暗币（Darkcoin）发布于 2014 年 1 月。作为匿名货币，它对所有交易采用一种叫作 DarkSend 的再混合协议。另外，黑暗币的出名还在于它在工作量证明算法中使用了 11 种不同的哈希函数（blake、bmw、groestl、jh、keccak、skein、luffa、cubehash、shavite、simd、echo）。

提示：块产生时间，2.5 分钟；总货币量，2200 万（2016 年年中）；共识算法，作业证明多算法共用；市值，1900 万美元（2014 年年中）。

2.3.3 货币的未来

加密货币的未来整体来说比比特币更加繁荣。它们在比特币的基础上引进了全新的去中心化的组织形式和共识机制，并由此衍生出了数以百计的不可思议的创新。这将影响与经济相关的众多部门，如：财政、经济、货币、中央银行、企业管理等。

许多之前需要中心机构来执行授权或信用控制的活动，现在可以去中心化了。区块链和共识机制的发明，在根除权力集中、腐败、监管俘获的同时，必将大幅度削减组织和大规模系统协调的费用。

2.4 本章小结

本章主要介绍了区块链技术的基础知识。首先，介绍了区块链技术，这是这本书的基础，包括区块链技术的基本概念、框架与特点，核心技术及交易流程。在这个基础之上，我们详细介绍了区块链技术最成功的应用——比特币，包括概念、原理、及隐私模型。之后，我们介绍了区块链的另外一个重要应用——以太坊。以太坊是区块链技术发展的一个重要的方向，是区块链技术未来的一部分。我们介绍了以太坊技术、以太坊智能合约、以太坊去中心化应用、以太坊发展的现状及未来。最后，我们分析了现有的流行的电子货币各自的优缺点。

参考资料

[1] 袁勇，王飞跃．区块链技术发展现状与展望 [J]. 自动化学报，2016，42(4).

[2] Antonopoulos A M. Mastering Bitcoin: Unlocking Digital Crypto-Currencies[J]. Oreilly Media Inc Usa, 2015.

[3] Swan M. Blockchain: blueprint for a new economy[M]. O'Reilly, 2015.

[4] 以太坊（Ethereum）：下一代智能合约和去中心化应用平台．以太坊爱好者，2016-06-23.

[5] Deutsch Bank, Banking & Technology Snapshot, DB Research, 20 December 2012.

[6] Vatalik Buterin, Merkling-in-Ethereum, https://blog.ethereum.org/2015/11/15/merkling-in-ethereum/.

第3章 区块链架构剖析

前面几章介绍了区块链背景以及相关的基础知识，本章将为读者介绍区块链的架构。区块链源于支持 BitCoin 虚拟货币系统的底层基础架构，在支撑 BitCoin 平稳运行三四年后，以其独特的去中心化架构逐渐吸引 IT 业界的关注，使得业界的关注点逐渐从虚拟货币转移到区块链平台上，并被认为是目前呼声最高的下一代互联网——“价值互联网”的颠覆性技术。

本章将深入剖析区块链基础架构（区块链 1.0），阐述其架构属性和特点，同时也详细分析从基础架构上延伸扩展的区块链 2.0 和区块链 3.0 架构，最后介绍用来集成整合不同区块链的互联链架构。

3.1 基本定义

在详细讨论区块链之前，为了便于准确地把握区块链的架构，我们先给出区块链的定义。由于目前在业界并没有统一的区块链定义，我们将用渐进逼近的方式来定义区块链，以求完整、准确。

定义 1：区块链

1）一个分布式的链接账本，每个账本就是一个“区块”；

2）基于分布式的共识算法来决定记账者；

3）账本内交易由密码学签名和哈希算法保证不可篡改；

4）账本按产生时间顺序链接，当前账本含有上一个账本的哈希值，账本间的链接

保证不可篡改；

5）所有交易在账本中可追溯。

该定义中“分布式”的定义如下。

定义 2：分布式

分布式是一种计算模式，指在一个网络中，各节点通过相互传送消息来通信和协调行动，以求达到一个共同目标[1]。

在区块链中，分布式包括“完全去中心”“部分去中心”和“部分中心”3 种模式，分别对应区块链的 3 种部署模式：“公共链”“联盟链”和“私有链”。分布式意味着在区块链网络中不存在一个中心节点，该节点负责生成、修改、保管所有账本。

定义 3：完全去中心

一种网络的架构模式，在该模式下网络没有拥有者，完全对外开放。网络中每个节点都可选择拥有相同的权限。在完全去中心的区块链网络上，所有节点都可以读写区块链数据，都可作为记账的候选节点参与共识流程，有机会参与账本的生成和记账。

定义 4：部分去中心

一种网络的架构模式，在该模式下网络属于一个联盟共同所有，网络只对联盟成员开放。网络中每个节点被赋予不同权限。在“部分去中心”的区块链网络上，节点根据所赋予权限读写区块链数据，参与共识流程以及参与账本的生成和记账。

定义 5：部分中心

一种网络的架构模式，在该模式下网络属于一个所有者，网络只对所有者内部成员开放。网络中每个节点被赋予不同权限。在“部分中心”的区块链网络上，节点根据所有者赋予的权限读写区块链数据，参与共识流程以及参与账本的生成和记账。

在有了一个明确的区块链定义之后，我们来看比特币下的区块链架构。在此之前，我们也对大家耳熟能详的“架构”一词做一个严格定义。

定义 6：架构

架构有两个层面的涵义。一个是静态层面的，主要是勾画系统边界、结构、组成的组件以及组件之间的关联关系；另一个是动态层面，主要是规范组件的行为以及组件之间的交互协议。根据一个 IT 系统的架构，可以界定该系统的功能特性和一些非功能特性[2]。

例如：对于 Bitcoin，它的功能可以是虚拟货币的发行和流通，是支付，结算和清算，保证不可“双花”、交易不可篡改、交易可追溯；非功能特性则包括安全措施（签名、

加密、隐私保护等）以及平均出块时间、每秒交易量等。

架构设计要考虑不断变化的和恒久不变的两方面。

一个有长久生命力的系统都有一个设计高明的架构，其精髓在于架构能支持系统功能的变化、发展、演化，允许系统功能不断变化，也就是架构必须提供的灵活性。

而系统对易用性、安全性、稳定性和性能却应该是恒久不变的，因此 IT 架构的设计必须强调非功能特性，其中开放性、可扩展性、可移植性、可维护性、灵活性、安全性、性能（响应时间、吞吐率、并发数等）最为重要。比特币出现短短几年后，从中衍生出多种代币和应用，可以窥见其架构设计的安全、巧妙、灵活和可扩展性。

区块链因应用场景的不同而有不同的架构。在《区块链：新经济蓝图及导读》一书中，作者 Melanie Swan 提出一个按区块链应用范围和发展阶段来划分的概念，把区块链应用分为区块链 1.0、2.0 和 3.0。她把比特币作为区块链 1.0 的典型应用。区块链 1.0 支撑虚拟货币应用，也就是与转账、汇款和数字化支付相关的密码学货币应用。区块链 2.0 支撑智能合约应用，合约是经济、市场和金融的区块链应用的基石。区块链 2.0 应用包括股票、债券、期货、贷款、抵押、产权、智能财产和智能合约。区块链 3.0 应用是超越货币、金融和市场的范围的去中心化应用，特别是在政府、健康、科学、文化和艺术领域的应用。

区块链也会因部署模式的不同而有不同的架构。如公共链、联盟链、私有链和侧链。多种区块链架构的出现也使得链与链之间的集成、整合成为挑战，因此互联链（Interledger）的概念也应运而生[3]。

下面我们逐个讨论不同架构的区块链。

3.2　区块链 1.0 架构：比特币区块链

区块链 1.0 的典型应用是比特币应用。比特币是第一个解决“双花”问题的去中心化虚拟货币系统。下面我们来详细剖析比特币系统的架构。

中本聪在发表比特币白皮书后的第 2 年，就发布了比特币的第一版实施系统。据普林斯顿大学出版的《 Bitcoin and Cryptocurrency Technologies 》作者们推测，中本聪很可能是先写好了比特币系统，才写比特币的白皮书的[4]。从最初的比特币源代码可以看出，比特币系统没有很明确的模块划分，很多不同功能都放在一个 5000 多行的 Main 程序中实现。因此，当初中本聪并没有从架构上考虑太多，而是用简单直白的办法一气呵成地把比特币系统写了出来。当参与开源项目的开发者多起来的时候，一个清晰的架构

就显得愈来愈重要，这样对代码的重用、维护和扩展非常重要。在 2013 年 12 月，一个将比特币代码模块化的建议《Post-0.9 modularization of Bitcoin Core》提交了上去。从目前发布的 Bitcoin 0.12 版本看，比特币的模块化工作还在进行过程中。因此，要很清晰地在代码层面理清比特币的架构还需要一个比较长的过程。

由于比特币是基于 P2P 架构的虚拟货币系统，因此它的架构和我们熟悉的分布式架构有很大不同。从一个客户端 / 服务端（Client/Server）、浏览器 / 服务端（BS 架构）或三层架构（3-Tier Architecture）等角度来看比特币架构，很容易感到困惑。比如说大多数比特币的软件，一般都叫客户端（Client），大家自然会想到一定有个服务端（Server）在后台。但其实在比特币里，服务端这个概念被弱化了，即使是看起来很像服务端的 bitcoind，也被称为没有界面的客户端。唯一被称为服务端的是在 bitcoind 里的 HTTP/JSON RPC 服务端。该组件只是用来提供对外 HTTP、JSON RPC 的服务接口。

图 3-1 是根据目前的代码情况勾画的比特币架构图。比特币架构总体上分为两部分，一部分是前端，包括钱包（Wallet）或图形化界面；另一部分是运行在每个节点的后台程序，包括挖矿、区块链管理、脚本引擎以及网络管理等功能。

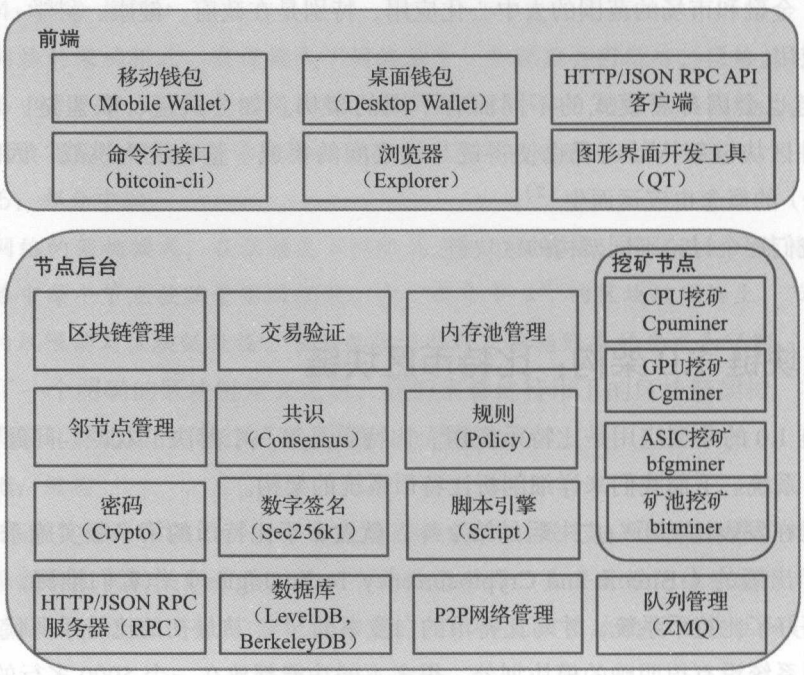


图 3-1　比特币架构

接下来将逐个介绍其功能。

3.2.1　比特币前端

1. 钱包

钱包保存用户的私钥数据库，并管理用户的余额，提供比特币交易（支付、转账）功能。

钱包分为两种：非决定性钱包和决定性钱包[5]。

1）非决定性钱包（Nondeterministic Wallet）：该类钱包直接保存私钥，私钥数据保存在 Berkeley DB 上。所谓决定性（deterministic），指的是私钥是否由种子生成。如果是就叫“决定性”，反之就叫“非决定性”。非决定性钱包因为直接保存私钥，如果私钥被盗窃，钱包里的比特币就会被盗走。因此非决定性钱包的安全性不高。另外如果私钥比较多，该类钱包管理起来就比较麻烦，备份也不方便。而且比特币系统出于提倡隐私保护的缘故，不鼓励地址重用，因此虽然比特币核心带有一个非决定性钱包，但目前并不被推荐使用。

2）决定性钱包（Deterministic Wallet）：目前建议使用的一种钱包。该类钱包所有的私钥都由一个私钥种子（Seed）通过单向哈希算法生成，因此备份该类钱包非常容易，只要备份私钥种子，就可以利用种子一次性恢复所有的私钥。决定性钱包又分两种类型，类型 1 是普通决定性钱包，其私钥种子直接生成所有私钥；类型 2 是层级决定性钱包（Hierarchical Deterministic Wallet），它的私钥保存在一个树形结构，由一个总私钥生成父私钥，父私钥生成子私钥等。

钱包从部署场景来说，分为 4 种，分别为：移动钱包、桌面钱包、互联网钱包，以及纸钱包。

移动钱包顾名思义是运行在智能手机、移动终端的轻量级钱包。该类钱包由于运行在资源有限的环境，一般不会下载整条区块链，而多数采用一种叫“简化支付验证”[⊖]（Simplified Payment Verification，SPV）的方法验证交易。这种钱包也叫 SPV 钱包。该方式依靠网络上的可信任节点查询所有区块的区块头，以及按照交易的确认数（相当于黑客攻击的难度），再有就是能否在相应的区块中找到该笔交易来验证支付的真伪。

⊖ 简化支付验证：用户不需验证交易，只是连接到一个可信任的节点，从该节点下载所有的区块头而不需要下载整条区块链。该节点要求对方发的交易符合特定的模板，例如交易地址、交易的 Merkle 树分支等。用户如果能够从区块链的某处找到相符合的交易，他就可以知道网络已经认可了这笔交易，而且得到了网络的多少个确认。

图 3-2 所示的比特币 Android 钱包，是一个基于 bitcoinj 实现的开源 Android 钱包，于 2011 年 3 月 7 日发布。它支持 Android 4.0 以上或黑莓 OS 10 以上版本。到 2015 年 6 月，已经有 80 万用户使用该钱包。

移动钱包的优点是灵活方便，缺点是因为不保存整条区块链，不做交易的全验证，因此安全性不是特别好。

桌面钱包也分两种，一种是厚钱包（Thick Wallet），另一种是薄钱包（Thin Wallet）。厚钱包下载整条区块链，并进行完整的交易校验。比特币核心（Bitcoin Core）就是一个厚钱包。图 3-3 所示是比特币核心钱包。它提供完整的钱包功能，包括签名、钱包加密、备份、密钥导入、导出等。另外像 armory 钱包，可以管理多个钱包，并将钱包离线保存，以防止攻击。薄钱包不下载整条区块链，而是采用 SPV 等方式来验证与用户相关的支付交易，例如 Multibit 钱包。另外像 Electrum 钱包也算薄钱包，它本身不使用 SPV 方式，但也不保存完整区块链，而是信任它的服务端的验证。

图 3-2 比特币 Android 钱包

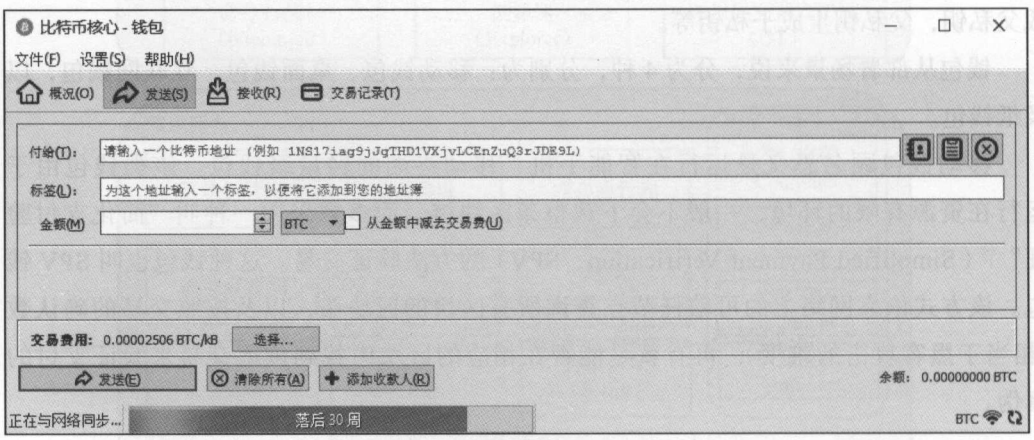


图 3-3 比特币核心钱包

厚钱包的优点是安全，缺点是有交易全验证的开销，适合于资金安全性要求高的场景，比如非小额支付等场景。薄钱包的优点是灵活高效，但安全性不高，适合于小额支付场景。

互联网钱包也不下载整条区块链。其优点是可以在任何地方、任何设备管理钱包。互联网钱包依托第三方平台提对用户隐私的保护。互联网钱包也和桌面薄钱包一样有安全性不高的问题，但使用起来更灵活方便。例如 Blockchain.info、CoinCorner 等。如图 3-4 所示是 Blockchain.info 提供的互联网钱包。

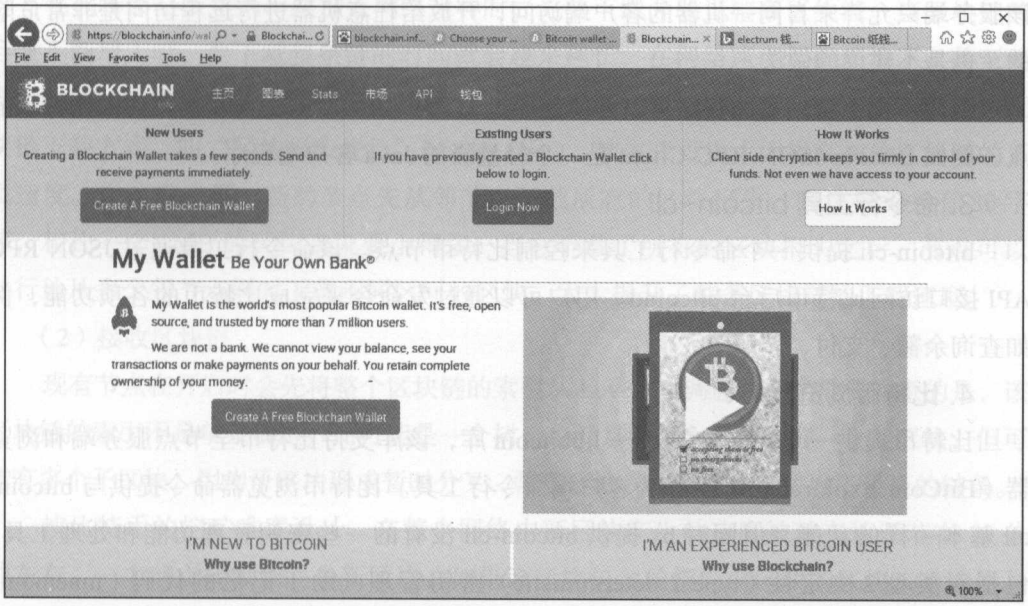


图 3-4　Blockchain.info 提供的互联网钱包

纸钱包用于将私钥进行冷备份，可以用于防范由于电脑或 USB 介质损坏所造成的私钥丢失。同时由于纸钱包离线存放，也能防范黑客通过网络攻击盗窃私钥，但也要防范纸钱包被人物理偷窃或复制。如图 3-5 所示为一个纸钱包，左边的二维码是比特币的地址信息，右边是用户的私钥二维码。

图 3-5　纸钱包

2. HTTP / JSON RPC API

比特币提供 HTTP / JSON RPC API 接口，供外部通过接口控制比特币节点。当运行比特币核心 bitcoin-qt 传入 -server 选项时，或运行 bitcoind 时，比特币提供一个 HTTP/JSON RPC 服务端，外部程序可以通过 JSON RPC API 来访问比特币节点。缺省情况下该服务端只允许来自同一机器的客户端访问，开放给任意机器进行远程访问是非常危险的，也是不建议的做法。

使用不同语言编写的程序可以方便地通过 HTTP / JSON RPC API 接口访问比特币节点。例如 Bitcoin-JSON-RPC-Client 是一个轻量级的 Java 客户端程序。

3. 命令行工具 bitcoin-cli

bitcoin-cli 提供一个命令行工具来控制比特币节点。该命令行工具通过 JSON RPC API 接口访问比特币后台 bitcoind。用户可以通过发命令来完成比特币的各项功能，例如查询余额、支付、转账等。

4. 比特币浏览器 bx

比特币提供一个跨平台的 C++ libbitcoin 库，该库支持比特币全节点服务端和浏览器（BitCoin Explorer（bx））作为客户端命令行工具。比特币浏览器命令提供与 bitcoin-cli 基本一样的功能。但同时 bx 提供 bitcoin-cli 没有的一些密钥管理功能和处理工具，包括对类型 2 决定性（type 2 deterministic）密钥管理、助于记忆的代码（mnemonic code）的密钥生成（例如用易于记忆的词汇作为私钥种子）、隐秘地址、支付和查询支持。

5. 图形开发工具（Qt）

比特币核心是比特币使用最广的客户端，它是使用 C++ 开源用户界面开发工具 Qt 所开发的桌面客户端。Qt 是一个跨平台的 C++ 图形用户界面应用程序框架。它提供给开发者建立图形用户界面所需的功能，广泛用于开发 GUI 程序，也可用于开发非 GUI 程序。Qt 是完全面向对象的，很容易扩展，并且允许真正的组件编程。Qt 支持更多的平台（包括 Microsoft Windows、GNU/Linux、Mac OS X、Android、iOS、WinCE、UNIX 家族等）。

3.2.2 比特币节点后端

比特币节点后台负责参与比特币网络的通信互联，维护区块链，验证区块、交易，广播、转播传递区块交易信息。比特币的后台程序主要是由 bitcoind，以及挖矿节点程序构成。比特币核心 bitcoin-qt 实际上是包含前后端（除挖矿功能以外）的一体化节点。下面简单介绍比特币后端组件的功能。

1. 区块链管理

区块链管理涉及初始区块链下载、连接区块、断开区块、校验区块和保存区块，以及发现最长链条的顶区块。区块链管理的代码逻辑都在 main.cpp 程序中实现。

（1）下载区块链

在比特币全节点第一次加入网络运行时，先要下载并验证整条区块链。当区块链的容量不断增大时，要下载整条链的时间就会越来越长，在网络速度低的环境下，甚至要几天时间才能完成整条链的下载。2014 年，比特币 0.10.0 版本正式发布一个新的初始区块链下载方式，叫“区块报头先行”(header first)。该方式可以大大提高初始区块链的下载速度。在该模式下，新的节点先从邻节点下载所有的区块报头，区块报头只有 80 字节，相比一个 1MB 的区块来说要小得多。当它下载了所有的区块报头之后，节点可以并行地从多个邻节点同时下载不同区间的区块，大大提升了整条区块链的下载速度。

（2）接收区块链

现有节点在开启时会先将整个区块链的索引从 LevelDB 调进内存。需要注意的是，该区块链的索引不是单条的链，而可能是一个树，也就是说每个区块只有一个父区块，但可能有多个子区块，因为子区块形成暂时分叉，需要逐渐发现哪个子区块属于最长的链条。

按比特币的挖矿难度设计，在整个网络中新区块产生的速度被动态地调节在 10 分钟左右。一个节点接收到一个新区块，如果该区块包含的指向前区块的区块头哈希值与节点当前顶端区块哈希值相同时，该节点会尝试连接新接收的区块，并将其作为当前链条的最顶端区块，从而延伸节点所拥有的区块链。这是因为在区块链中，每个区块都保存有前个区块的区块头哈希值，并通过这个方式来链接区块。

（3）区块链验证

在区块链管理中，连接区块函数 ConnectBlock() 是一个检测“双花”交易的关键。该函数对新接收的区块中的所有交易进行检测，验证是否每个交易的比特币来源都能在当前的“尚未花比特币”（Unspend Transaction Output，UTXO）记录中找到匹配。在网络延迟情况下，节点接收的区块可能不按顺序到达，在这种情况下，有些交易的比特币来源可能在 UTXO 记录中暂时找不到，但当后面收到延迟到来的区块后，UTXO 记录会被更新，区块链条会链接起来。举个例子说明就会比较清楚，假设节点当前的区块链顶端区块是 B1，由于网络延迟问题它先收到 B3，在验证 B3 中的交易的时候，有些交易的比特币来源可能在当前的 UTXO 记录中找不到。但当后面接收到 B2 后，UTXO 记录会被更新，如果 B2 和 B3 的交易都被校验后，当前的区块链条也从 B1 延展到 B2、B3，同时也会建立回滚记录并存放在磁盘中。回滚记录将用于断开现有区块。区块校验的流程图如图 3-6 所示。

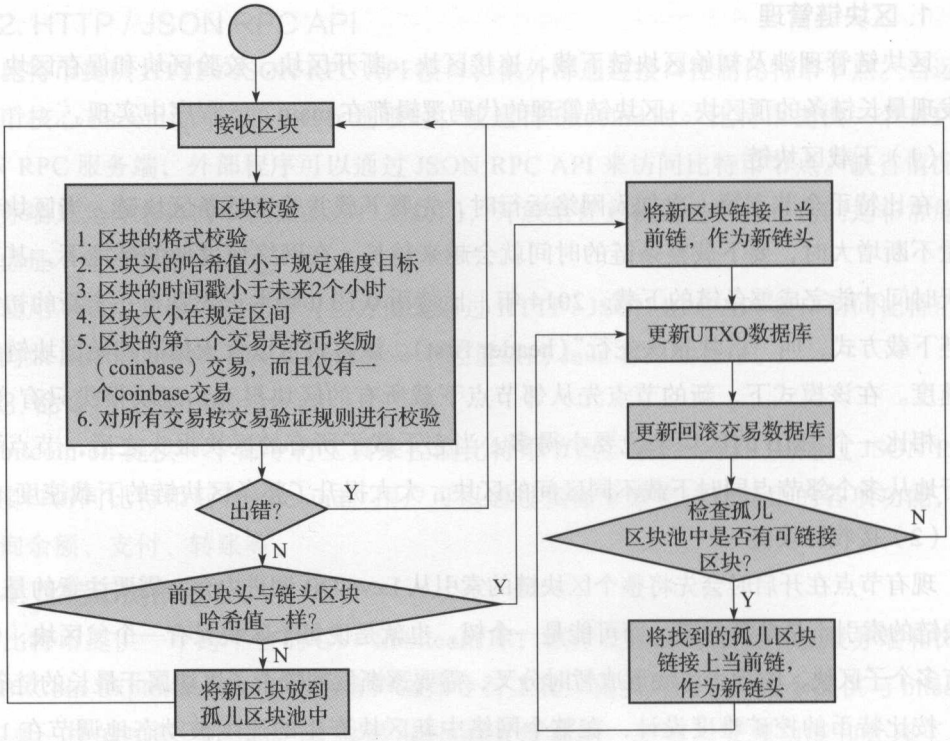


图 3-6 区块验证流程图

（4）重组区块链

当节点发现网络中有一条不基于它当前区块链的一条更长区块链时，它需要断开现有的区块并对区块链进行重组。大部分重组只是一个区块的重组，多发生在不同矿工几乎同时挖到合法的区块时。比如有 A 和 B 区块，有些网络节点先接收 A，有些节点先接收 B，其后的链条会发生分叉，假如有 B 区块的链条变得更长，有 A 区块的网络节点都会很快通过断开 A、连接 B 来重组它的区块链。断开区块、重组区块链涉及 UTXO 更改，被断开的区块中交易会回退到交易内存池（mempool），这个时候“回滚”记录就可以用来回滚断开区块中的交易。

2. 区块验证

交易验证模块会基于以下条件检查收到的比特币交易是否合规：

- 交易的格式是数据结构必须正确；
- 交易的输入和输出不能为空；
- 交易的大小不能超过定义的区块最大值 MAX_BLOCK_SIZE；
- 每个交易的输出，以及所有输出的合计，必须在一定范围内，也就是大于 0，小

于 2100 万比特币；

- 交易输入的哈希值不能为零，不应该转播挖矿（coinbase）交易；
- nLockTime 小于等于 INT_MAX；
- 交易的字节大小要等于或大于 100；
- 交易的签名操作数要小于签名操作的上限；
- 解锁脚本（scriptSig）只能把数字放入堆栈，锁定脚本（scriptPubkey）必须是标准格式；
- 和收到的交易相匹配的交易必须能在当前交易池或是主链上某个区块中找到；
- 对交易的每个输入，如果其对应的 UTXO 输出能在当前交易池中找到，该交易必须拒绝（双花交易），因为当前交易池是未经记录在区块链中的交易，而交易的每个输入应该来自确认的 UTXO，如果在当前交易池中找到，那就是双花交易；
- 对交易的每个输入，如果其对应的 UTXO 输出不能在主链或当前交易池中找到，该交易是一个孤儿交易，应将该交易放入孤儿交易池中；
- 对交易中的每个输入，如果其对应的 UTXO 输出是一个挖矿初始（coinbase）交易，该初始交易应该获得 100 个确认区块的确认；
- 对交易中的每个输入，其对应的输出必须是 UTXO（存在且没被花掉）；
- 用对应的输出交易来获得输入的值，检查每个输入的值及其合计，应该在允许的区间（0 ~ 2100 万比特币）；
- 如果一个交易的输入合计小于输出总计，则拒绝该交易；
- 如果交易的费用太低，则拒绝该交易；
- 每个输入的解锁脚本（unlocking script）必须和相应输出的锁定脚本（locking script）共同验证交易的合规性。

最后一个检查可能不容易理解，但却是比特币平台设计的精髓。比特币的一个很大的创新是依靠脚本来验证交易的合法性，即每一个将要花掉的比特币必须有相应的来源。简单来说，比特币交易中的输入（input）和输出（output）都由脚本和数值组成。通过比特币的脚本引擎，在一个简单的堆栈式计算平台上执行。举个例子说明可能帮助理解。一个交易的输入部分的解锁脚本是 <sig> <pubKey>，而与其对应的 UTXO 的锁定脚本是 OP_DUP OP_HASH160 <pubKeyHash?> OP_EQUALVERIFY OP_CHECKSIG。在最初的比特币实现中，是将解锁脚本和锁定脚本串起来一同执行，如果最后结果是逻辑值“真”（TRUE），则验证该交易的比特币来源是合法的。图 3-7 展示了脚本执行的顺序和每个脚本执行后的堆栈状态。具体执行过程如下：

1）由于是堆栈式计算引擎，因此作为数值的 <sig> 和 <pubKey> 相继入栈。当执行脚本 OP_DUP 时，它将堆栈头的 <pubKey> 复制一份也压入堆栈。

2）脚本 OP_HASH160 执行，将把堆栈头的 <pubKey> 弹出，并用 HASH160 算法进行哈希处理，哈希结果放回堆栈。然后遇到 <pubKeyHash?> 数值，该数值也被压入堆栈。

3）脚本 OP_EQUALVERIFY 把堆栈头端的两个哈希值都弹出，并进行比较。如果不一样，验证就出错，交易不合法。如果验证通过，堆栈只剩下 <sig> 和 <pubKey>，最后脚本 OP_CHECKSIG 将两者弹出，执行检查签名的脚本，验证该交易签名是否是由拥有该公钥对应用户用其私钥签的，如果是，交易就是合法交易，否则就是不合法交易。

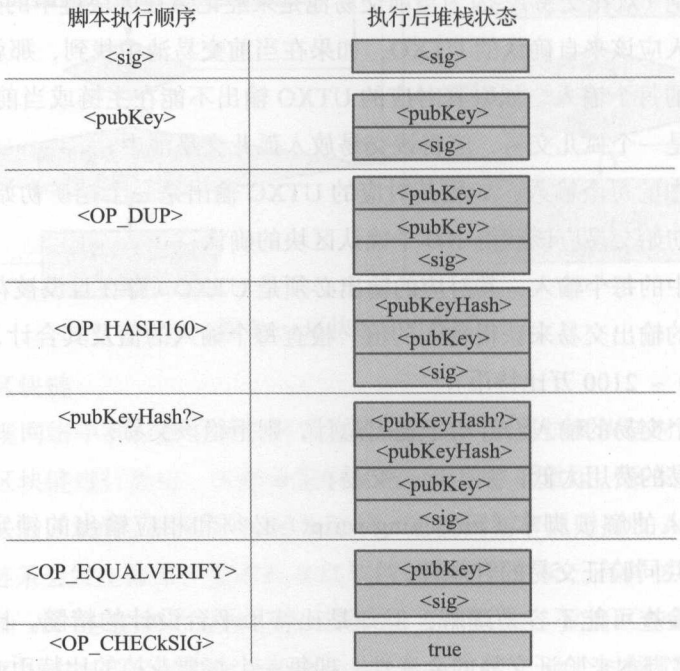


图 3-7 比特币脚本交易验证示意

由于将解锁脚本和锁定脚本串起来一同执行会产生安全风险，为防止恶意的解锁脚本将数值压入堆栈破坏锁定脚本，从 2010 年开始，不再将解锁脚本和锁定脚本串起来执行，而是分开执行，执行的结果再复制到锁定脚本执行的堆栈。

3. 内存池管理

比特币内存池（mempool）管理也就是交易池管理。节点将通过验证的交易放在一个交易池中，准备放在一个挖到的区块中。当矿工挖到一个合格的区块后，他将按一定

的优先级次序从交易池中选出交易放到区块中。优先级是按交易中的输入对应的 UTXO 的“链龄”和交易额的大小来划分的。越老 UTXO 的交易以及交易额越大的交易优先级会越高。优先级（Priority）采用以下公式计算：

$$\text{Priority}=\text{Sum}(\text{Value of input}\times\text{Input Age})/\text{Transaction Size}$$

在此公式中，输入的值是按比特币基础单位（satoshi）计算，1 个 satoshi 等于一亿分之一个比特币。UTXO 的链龄按已在链上记录该交易的区块为起点，按后面有多少个区块来计算，也就是计量该区块在区块链的“深度”。交易的大小以字节为单位。要成为高优先级交易，一般来说优先级值要高于 57 600 000，这个优先级相当于 1 个比特币交易额的交易，已有一天（24 小时）的“链龄”（相当于已有 144 个区块的确认），交易长度有 250 个字节。比特币区块中的前 50K 字节保留给高优先级的交易。高优先级的交易费即使是零都会被包括进区块链的优先区域。

当区块填满后，剩下的交易会留在内存池，等待下一个区块的到来。随着它们的“链龄”的逐渐增加，它们以后被选中的几率也会逐渐增加。在内存池的比特币的交易不会过期，对于一个交易费用为零的交易，最终也会因为其对应的 UTXO 链龄的逐渐增加而被包括在区块链上。但是内存池的交易不保存在硬盘上，当挖矿节点重启时，内存池的交易会被清空。所以也会出现一种情况，就是一个低优先级的交易由于挖矿节点在网络频繁离开而不存在于任何挖矿节点的内存池中。因此，如果在一定时间内一个交易一直不能被矿工包括在区块链上，钱包软件需要重新发送该交易，并附上较高的交易费。

在一些比特币节点的实现也维护一个独立的“孤儿”交易池。如果一个交易的输入相对应的 UTXO 不能被找到，也就是没有“父”交易，会被当作“孤儿”交易，暂时放在“孤儿”交易池。当父交易来到后，该“孤儿”交易会被从“孤儿”交易池移到内存池。

4. 邻节点管理

当一个新比特币节点做初始启动（bootstrap）的时候，它需要发现网络中的其他节点，并与至少一个节点连接。一般是与一个已知的节点在 8333 端口建立 TCP 连接。连接的“握手”流程发送一个版本信息，包括：P2P 协议版本，本节点支持的服务，当前时间，对方节点 IP 地址，本节点 IP 地址，比特币软件版本，以及本节点当前区块链的长度。对方节点收到握手信息后会回复一个收到确认的信息。

那么新节点如何发现邻节点呢？

第一个办法是用一些“ DNS 种子”查询 DNS。“ DNS 种子”是提供比特币节点地

址的 DNS 服务器。比特币核心带有 5 个不同的“ DNS 种子”。DNS 种子可提供稳定的比特币节点地址。

第二个办法是直接把一个已知的邻节点作为种子节点，然后通过它发现更多的邻节点。当发现新的邻节点后，新节点一般将断开和种子节点的连接。新节点将其地址信息发给邻节点，邻节点会继续将新节点的地址转发给它们的邻节点，这样新的节点会在网络上被其他节点知道，并保持其在网络上连接的畅通。另外新节点将发 getaddr 信息给邻节点，邻节点收到后将把所知的地址信息发送给新节点。新节点一般会维持与 8 个邻节点的连接。新节点启动结束后，它会记住最近连上的邻节点的地址。当它重新启动的时候，它能够很快地完成和已知邻节点的连接。如果以前的邻节点都连不上，它会重新开始初始启动流程。用户也可以通过提供一个固定 IP 地址列表来替换比特币系统自动管理邻节点的流程。

如果一个连接上一段时间内没有信息交互，节点会定期发一些信息去维护连接。如果一个节点和邻节点的连接在超过 90 分钟里没有联系，该邻节点会被认为下线，节点会寻找一个新的邻节点来进行连接。因此，无需要中心控制，网络节点可以自由加入或离开网络，即比特币网络能动态地调节节点的连接，以保证比特币网络的正常运行。

5. 共识管理

比特币里广义的共识管理（Consensus）应该包括挖矿、区块验证和交易验证规则。但这些功能实现目前分散在不同的程序中。由于比特币的关键是在陌生 P2P 环境建立共识机制，因此共识管理至关重要。目前比特币的开源社区也意识到共识管理的模块化非常重要，因为不同的比特币实现可以简单重用共识管理模块，以保障验证规则的一致性。未来共识模块将从比特币核心中分离出去，作为独立的模块。这样未来共识逻辑的改变也不会对比特币核心带来影响。目前在比特币 0.12.0 版本中，一部分共识管理的代码已经移到 consensus 子目录。可以预见，未来更多的共识管理逻辑将移到该子目录，成为可插拔的共识模块。

比特币的共识管理必须支持前向兼容，即使过去版本有缺陷（bug）也要保持，否则比特币网络会出现分叉。

目前在 consensus 子目录的共识程序有 consensus.*、merkle.*、params.* 和 validation.*。

6. 规则管理

比特币的共识规则是所有节点都必须遵守的规则（policy）。而每个节点可以采用一些共识规则以外的个性化规则。这部分的规则由规则管理模块实现，目前放在 policy 子目录中。一个个性化规则的例子是将交易存放内存池的规则。比如一个节点可以拒绝保

存、中转大于 200KB 的交易。这个规则和共识规则不会产生矛盾。如果一个实施该规则的节点，收到一个包含大小为 200KB 的交易的区块，它不会拒绝该区块，只不过它不会保存或中转大小为 200KB 的交易。另外像对交易费用的一些规则，也可通过规则模块来管理。

7. 密码模块

密码模块（Crypto）主要是处理比特币地址，采用 RIMEMD 和 SHA-256 算法以及 Base-58 编码来生成比特币地址。目前代码放在 crypto 的子目录中。比特币的公钥是通过私钥产生的，然后采用 Secure Hash Algorithm（SHA）算法 SHA256 和 RACE Integrity Primitives Evaluation Message Digest（RIPEMD）算法 RIPEMD160 对公钥进行处理，最后通过 Base58 编码形成比特币地址。例如对一个公钥 K，处理流程先进行 SHA256 哈希处理，然后再对哈希值进行 RIPEMD160 哈希处理，得出一个 160 位（20 个字节）的结果，最后经过 Base58Check 编码，形成可读的字符串地址。熟悉技术的读者可能对 Base64 编码比较了解，Base64 编码主要用于基于文本的系统传送二进制数据，例如在邮件里发的附件都要进行 Base64 的编码。Base58Check 采用 Base58 编码，同时加入校验码，以防止出现不小心写错地址的情况。Base58 是 Base64 的子集，过滤了一些容易引起混淆的字符，例如 0（数字零），O（大写 o），1（小写 L），I（大写 i），以及 "+" 和 "/" 符号。

Base58Check 的校验码对地址信息进行双重 SHA256 哈希处理，并取前 4 位做校验码，加在比特币地址的后面，因此比特币地址带有校验信息，可以防止人为错误。

8. 签名模块

比特币采用椭圆曲线数字签名算法（ECDSA）来实现数字签名以及生成公钥。ECDSA 是一种非对称加密算法，是基于椭圆曲线离散对数问题的计算困难性的一种公钥密码的方法。secp256k1 曲线是由国际上最著名的椭圆曲线密码技术公司 Certicom 所推荐的椭圆曲线，具有比其他曲线更高的性能。原来大部分的比特币 secp256k1 ECDSA 实现是依赖于 OpenSSL 库，后来由于 OpenSSL 库可能会带来共识模块的 bug，因此在 2015 年以后，比特币开源社区发布了基于 secp256k1 的 ECDSA 专有实现 C 库，并放在 secp256k1 子目录中。

9. 脚本引擎

比特币的脚本语言是一种专门设计的、与“Forth”类似的、基于堆栈的编程脚本（script）语言。基于堆栈的语言的指令只按顺序执行一次，也就是说没有循环或跳转指令。因此脚本的指令数给我们一个程序运行时间的上限和所需的内存上限。这种基于堆

栈的运算平台不是图灵完备的运算平台。

提示：图灵完备是一个术语，当一组数据操作的规则（一组指令集、编程语言，或者元胞自动机）满足任意数据按照一定的顺序可以计算出结果，被称为图灵完备（turing complete）。元胞自动机（Cellular Automaton）是一种时间和空间都离散的动力系统，由冯·诺依曼在20世纪50年代发明。散布在规则格网（Lattice Grid）中的每一元胞（Cell）取有限的离散状态，遵循同样的作用规则，依据确定的局部规则同步更新。大量元胞通过简单的相互作用而构成动态系统的演化。图灵完备概念比较抽象，其含义不好理解，这里解释一下。简单说是指一个与通用图灵机计算能力相当的计算系统，一般来说在可计算理论中，一个计算系统如果能模拟单带图灵机，则被称为图灵完备。

图灵模型看似简单，实际上功能非常强大，通过图灵机的组合，可以解决世界上大部分的计算问题（除了图灵“停机问题”以及与其等价的计算问题以外。有兴趣的读者可以参考Tom Stuart著的《计算的本质：深入剖析程序和计算机》）。现代计算机归根结底都基于图灵模型，可以说图灵奠定了现代计算机科学的基础。

前面说过，比特币的脚本语言不是图灵完备的语言，它没有能力计算任意带复杂功能的任务，这也是比特币专门设计的，因为矿工必须运行这些脚本，运行者可以是来自任意参与网络的人，我们不希望矿工有能力提交一个可能有死循环的脚本，这也是从安全角度的设计。作为一个虚拟货币系统，这样的设计已经足够满足需求，同时还有很大的空间去扩展能力。而另一方面，以太坊的设计从一开始就是一个基于区块链的编程平台，不仅仅局限于虚拟货币，因此以太坊的虚拟机（EVM）支持图灵完备的编程语言。但以太坊会用Gas（燃料）机制来防止出现死循环的脚本，也就是说，提交脚本必须提供一定的Gas，相当于必须有少量的以太币来保证脚本的运行，当Gas用完，智能合约里的脚本也就不能继续执行。

比特币的脚本语言非常小，只能有256个指令，每个指令是一个字节长。这256个指令中，75个是保留指令，15个已废弃。但比特币通常使用的指令不多，见表3-1。很多指令在比特币网络上不能执行，因为每个节点可以有自己的脚本白名单，只允许运行白名单上的指令。

表 3-1　常用比特币脚本指令表

OP_DUP	将堆栈头上的内容复制一份并压入堆栈
OP_HASH160	弹出堆栈头内容，先用 SHA256 对其做哈希处理，再用 RIPEMD-160 对结果做第二次哈希处理，结果压入堆栈
OP_EQUALVERIFY	弹出堆栈头的两项内容，如果两个内容一样，返回“真”值；否则返回“假”值
OP_CHECKSIG	用输入的公钥检查输入的签名，如果签名符合，返回“真”值，否则返回“假”值
OP_CHECKMULTISIG	用提供的多个公钥检查多重签名的正确性

脚本引擎是校验交易的运算平台，从对解锁脚本（unlocking script）和锁定脚本（locking script）的自动执行校验可以看出该引擎的重要作用。另外脚本引擎也可以用来实现合约。像以太坊的虚拟机（EVM）实际上可以看成是比特币脚本引擎的扩展，它增加了图灵完备的运算能力。因此，新版本的比特币将脚本引擎放在 script 子目录中，将来可以变成可插拔引擎，使得引入新的功能更强大的引擎更为方便，不影响原有比特币的代码。

目前 script 子目录里有 interpreter.*、script.* 和 standard.* 程序。script.h 定义了脚本命令，interpreter.cpp 解析、评估和较验脚本命令，standard.h 定义了标准的交易。

10. 挖矿

比特币核心不带挖矿（mining）功能。比特币最早的挖矿程序是 cpuminer，是通过 CPU 来挖矿的。随着加入比特币的矿工增多，很快 CPU 就不能胜任挖矿所需的计算能力，GPU 取代 CPU 成为挖矿的主力，具代表性的程序是 cgminer。随着比特币价值的升高，比特币挖矿成为有利可图的业务。这时采用现场可编程逻辑门阵列 FPGA（Field Programmable Gate Array）和专用逻辑电路（Application Specific Integrated Circuit）的专业设备相继出现，到现在，基本上 CPU、GPU、FPGA 的挖矿设备相继退出历史舞台，目前使用的是清一色的 ASIC 挖矿设备。因为比特币的挖矿算法是 SHA256 算法，因此针对该算法优化的 ASIC 挖矿设备比其他挖矿设备有着性能上无可比拟的优势。Bfgminer 程序支持 FPGA 和 ASIC 挖矿设备，是目前比较流行的挖矿程序。随着比特币挖矿产业的发展，比特币挖矿池成为主流。挖矿池由矿池主控制挖矿，矿工参加集体挖矿，挖到的比特币按参与集体挖矿的矿工提交的工作量证明来分配。具体说来是矿工需按矿主需求提交工作量证明，这些工作量证明不一定是挖到的合乎比特币难度的区块，而是接近难度要求的区块。参与矿池挖矿获得分成的几率要大于单独挖矿的几率，而且矿池越大，计算力越强，挖到比特币的几率就会越大。因此比特币的矿池也越来越集中，目前大部分的矿池都是中国的矿池。中本聪当初设计比特币的目标是建立一个完全去中心化的虚

拟货币，采用“一个 CPU 一票”的理念。但后面出现比特币矿池高度集中、计算力中心化的问题却是中本聪始料未及的。矿池挖矿程序中比较流行的是 Bitminer。

很多人觉得比特币挖矿很神秘，其实简单说就是：不断对区块报头进行哈希处理，每次尝试改变一个随机数，直到区块报头的哈希值符合一定的条件，比如说起始必须有多少个零，才算挖到一个合格的区块。由于哈希处理不可逆，也就是说根据哈希算法得出的结果，不能反推出输入值，因此不能预知输入的参数，只能随机地试。而且两个不同的输入经哈希处理后得到的结果相同的几率小得可以忽略，因此比特币网络中谁能挖得到矿是一个随机的事件，几率的大小取决于节点的计算能力。

比特币挖矿的难度目标决定了网络大约多长时间能挖到一个块。比特币的设计目标是平均大约 10 分钟出一个区块。这个节奏决定了比特币的发币频率以及比特币交易的速度。图 3-8 显示的是比特币区块链高度为 421133 的区块。上面的难度位数值是 402990845，转成十六进制是 0x180526FD。比特币的难度目标是以十六进制数的前两位做指数，其余位数做系数，由下面公式计算出来：

$$\text{target} = \text{coefficient} \times 2^{(8 \times (\text{exponent} - 3))}$$

区块 421133 的系数是 0x0526FD，指数是 0x18，根据公式计算的结果是：

$$\text{target} = 0\text{x}0526\text{FD} \times 2^{(8 \times (24 - 3))} = 337661 \times 2^{168}$$

$$= 0\text{x}0000000000000000526FD0000000000000000000000000000000000000000$$

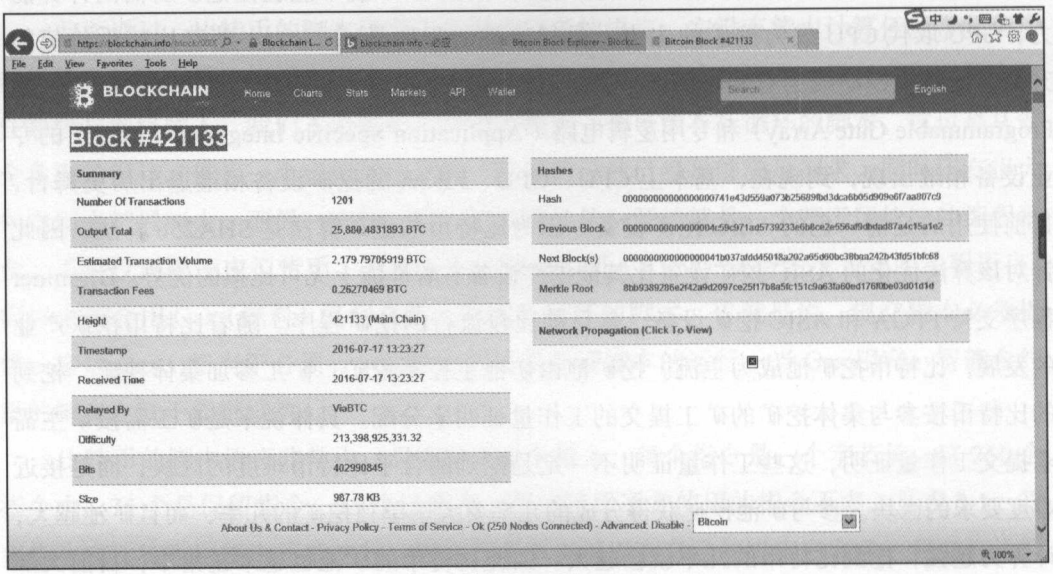


图 3-8　比特币区块信息

这是一个非常大的数，一般的计算器都会溢出。转成十六进制后，前 60 多位都应该是零。比特币的区块的哈希值要小于这个目标，才能算挖到一个合格的区块。

由于在网络上节点可自由出入，而且从趋势来说，整个网络的计算能力保持增长的态势。那如何把网络的出块速度稳定在 10 分钟一个呢？比特币网络是通过调整挖矿难度的目标来达到这个目的的。具体说来，就是每隔 2016 个区块，所有的节点都要重新更新比特币的挖矿难度目标。它挖到前 2016 个块所需要的时间与 20160 分钟（基于 10 分钟出块速度的两周时间）相比，如果前者时间短，比特币网络将调高难度，如果前者时间长，比特币网络会调低难度。

新难度计算公式是：

New Difficulty = Old Difficulty × (Actual Time of Last 2016 Blocks / 20160 minutes)

这样的话，区块的产出速度多会稳定在大约每 10 分钟一个块。

11. HTTP/JSON RPC 服务端

比特币在启动的时候，初始化程序 init.cpp 会启动 HTTP/JSON RPC 服务端的线程组。该组件对外提供 HTTP 和 JSON RPC 的接口，外部程序可以通过 JSON RPC 接口来调用比特币的 API，达到控制比特币节点的功能。该接口缺省是仅接收来自本机的客户端的连接请求。远程连接有极大的风险隐患，因此一般不推荐。读者可以参考网页 https://en.bitcoin.it/wiki/Original_Bitcoin_client/API_calls_list 上列出的最新 API 列表。

12. Berkeley DB 和 LevelDB 数据库

比特币用 Berkeley DB 做钱包数据库。Berkeley DB 是一个开源的文件数据库，介于关系数据库与内存数据库之间，使用方式与内存数据库类似，它提供的是一系列直接访问数据库的函数，而不是像关系数据库那样需要网络通信、SQL 解析等步骤。Berkeley DB 是一个轻巧而又性能高的嵌入式数据库，Berkeley DB 可以保存任意类型的键 / 值对，而且可以为一个键保存多个数据。Berkeley DB 可以支持数千个并发线程同时操作数据库，支持最大 256TB 的数据。

很多人在编译 BitCoin 软件的时候，会遇到 Berkeley DB 相关的错误。这个需要在操作系统中先安装 Berkeley DB 4.8。有些操作系统，例如 Ubuntu Kylin15.04 版本，需要配置 Berkeley DB 的源。碰到类似问题的读者可以尝试下面命令来修正错误：

```
sudo apt-get install libboost-all-dev
sudo add-apt-repository ppa:bitcoin/bitcoin
sudo apt-get update
sudo apt-get install libdb4.8-dev libdb4.8++-dev
```

比特币的区块原始数据不存放在数据库中，而只是作为文件类型存放在硬盘上。在系统的目录是 blocks/blk*.dat. 这个主要是用来快速搜索钱包缺少的交易记录，或者重组本地区块链的不同部分，同时给外部的节点提供区块链的同步服务。

LevelDB 用来存储区块的索引和 UTXO（未花的比特币交易输出）记录。LevelDB 是一个 Google 实现的非常高效的键值（Key Value）数据库，目前的版本 1.2 能够支持几十亿级别的数据量。在这个数量级别下还能有非常高的性能，主要归功于它的良好的设计。

在 blocks/index/* 这个目录中，存放着 LevelDB 中的已知区块的元数据，以及如何在硬盘找到它们的索引。没有这些索引文件，查找一个区块将会非常慢。

在 chainstate/* 这个目录中，存放着 LevelDB 中的 UTXO 记录，以及一些这些交易来源的元数据。这些数据用来校验收到的区块和交易。

另外回滚交易记录作为原始数据存放在 blocks/rev*.dat。在区块链分叉重组的时候需要用回滚记录去更新 UTXO 记录。

需要注意的是，LevelDB 的数据是冗余数据，可以用原始区块链数据来重建。但重建会花很长时间。但如果没有 LevelDB 的数据，比特币的校验和其他操作都会变得非常缓慢。

13. P2P 网络管理

P2P 网络管理的代码主要是在 P2P 网络上实现和其他邻节点的通信功能。这些通信功能包括：发现邻节点；连接并管理与邻节点的 Socket 连接；与邻节点交换不同的 P2P 消息（这些消息包含区块和交易）；有时在特殊情况下，会禁止异常行为的邻节点的连接。比特币节点的缺省配置是主动连接 8 个邻节点，同时允许最多 125 个其他节点发起的连接请求。比特币防止拒绝服务攻击（DoS）的方法主要是禁止异常行为邻节点的连接。如果一个邻节点传送非常明显的错误信息，该连接将被断开，而且其 IP 地址会被禁止，其重新连接申请会被拒绝。

大部分的 P2P 代码集中在 net.h/net.cpp。邻节点 IP 地址管理代码是 addrman.h/addrman.cpp。地址管理程序把地址存放在 peers.dat 数据库中，在启动的时候再把它调入内存。

比特币节点按功能分有几种。一种是全功能节点。全功能节点带有钱包、RPC 服务端，具有挖矿功能和进行节点校验区块和交易，并把区块和交易中转给与之相连接的邻节点。全功能节点会从头开始校验区块链，虽然它也可以采用区块链文件剪枝的方式来清掉区块链上一部分老的内容以减少硬盘空间的占用。另一种节点叫“基础全节点”，该种节点也做区块和交易的交易和中转，但不挖矿，不带钱包和 RPC 服务端。

还有一种比特币节点叫 SPV 节点，SPV 是 Simplify Payment Verification 的缩写，前面在介绍钱包时简单介绍过 SPV 的概念。SPV 节点信任别的节点来对区块和交易作校验。具体过程描述如下：

SPV 节点一般会在与邻节点的连接中设置过滤器，在比特币中叫 Bloom 过滤器，该过滤器只接收包含钱包里的公钥地址的交易。当一个邻节点看到一个交易与 SPV 节点的过滤器设置的条件符合，它就会用 merkleblock 消息来给该 SPV 节点发送一个区块。该 merkleblock 消息包含区块头和一个连接该交易到 Merkle 树根的一个 merkle 路径。SPV 节点可以利用该 merkle 路径来把交易和包含交易的区块联系起来，并验证交易包含在区块中。SPV 节点也用区块头来验证包含交易的区块和区块链中的其他区块能连上。这两个验证可以证明交易是记录在区块链上。总之，SPV 节点只需要接收小于 1KB 的区块头和 merkle 路径数据，这个是全节点接收数据的千分之一（全节点目前接收大小 1M 的区块）。

比特币 P2P 消息结构定义以及传输过程中的序列化 / 非序列化（serialization/deserialization）代码目前集中在 Main.h/Main.cpp 文件中。这些其实应该独立出来，因为该部分是比特币最基础的代码，可以给不同的模块重用。但现在的架构还不够理想，未来可能会重新组织该部分代码。

14. ZMQ 队列管理

比特币采用 Zero MQ 作为消息队列管理和消息分发工具。ZMQ 是一个简单好用的传输层，提供像框架一样的一个 socket library，它使得 Socket 编程更加简单、简洁，性能更高。它是一个消息处理队列库，可在多个线程、内核和主机盒之间弹性伸缩。

与很多人熟悉的 RabbitMQ 相比，ZMQ 并不像一个传统意义上的消息队列服务器，事实上，它也根本不是一个服务器，它更像一个底层的网络通信库，在 Socket API 之上做了一层封装，将网络通信、进程通信和线程通信抽象为统一的 API 接口。

3.3　区块链 2.0 架构：以太坊区块链

比特币的区块链架构主要围绕支持虚拟货币的实现，虽然它有一定的灵活性，但用来支撑虚拟货币以外的应用场景还显得非常局限。近年来，区块链逐渐引起 IT 业界的关注，并逐渐成为独立于比特币的一个平台架构，其重要性越来越受到重视。区块链 2.0 的概念也随之产生。其核心理念是把区块链作为一个可编程的分布式信用基础设施，支撑智能合约应用，以与过去比特币区块链作为一个虚拟货币支撑平台区别开来。具体

说来就是，不仅仅把区块链作为一个去中心化的虚拟货币和支付平台，而是通过增加链上的扩展性功能，把区块链的技术范围扩展到支撑一个去中心化的市场，交易内容可以包括房产的契约、权益及债务凭证、知识产权，甚至汽车、艺术品等。

区块链 2.0 提供一套新的协议（区块链 2.0 协议）支撑新型的去中心化应用。如果用互联网协议来做类比，区块链 1.0 就相当于 TCP/IP 协议，而区块链 2.0 就相当于 HTTP、SMTP 和 FTP 等高级协议。甚至有把区块链 1.0 比做电话，而区块链 2.0 相当于智能电话的比喻。在比特币后，出现很多被称为区块链 2.0 的平台，其中，最具代表性的是以太坊平台。下面简单介绍一下以太坊架构。

以太坊的设计主要还是以比特币架构为基础。前面几章已经介绍了以太坊的基本架构，本章不再详细叙述，下面只对和比特币架构不同的几个主要方面做重点讨论。以太坊架构图 3-9 所示。

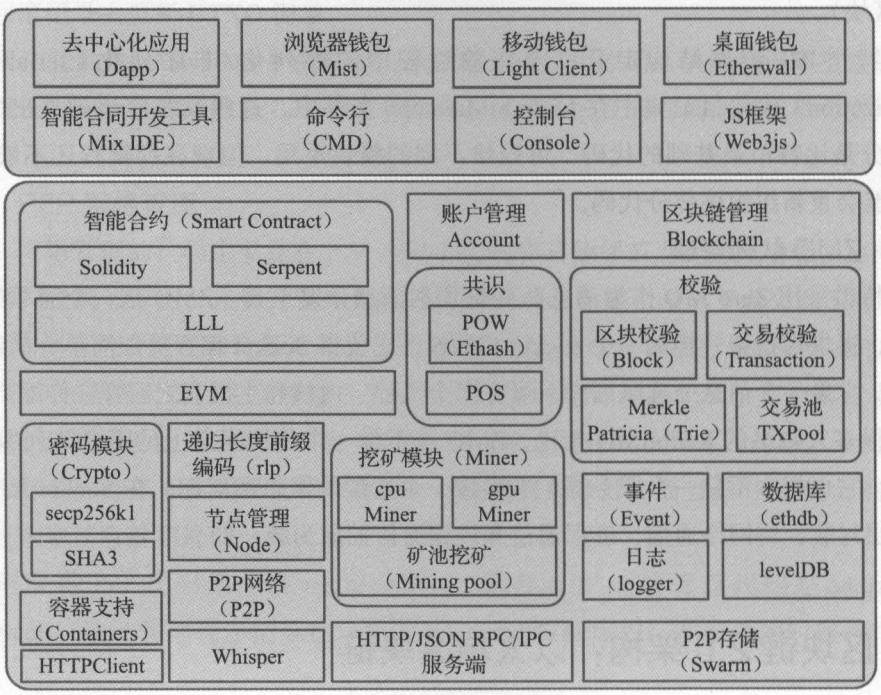


图 3-9 以太坊架构

1. 账户设计

比特币没有账户的概念。每个用户的余额都是从他们在区块链上的 UTXO 计算出来的。以太坊则有两种类型的账户：一种是外部所有账户（EOA），另一种是合约（Contract）账户。外部所有账户就是我们一般意义上的用户账户，它由私钥控制。合

约是一种特殊的可编程账户，合约存在以太坊区块链上，它是代码（它的功能）和数据（它的状态）的集合。合约受代码控制并由外部所有账户激活。

以太坊的设计是将区块链作为一个通用的管理对象状态转换的去中心化平台，账户就是有状态的对象。外部所有账户的状态就是余额，而合约账户的状态可以是余额、代码执行情况，以及合约的存储。以太坊网络的状态就是所有账户的状态，该状态由每个区块的交易来更新，同时需在全网形成共识。用户和以太坊区块链的交互需要通过对账户的交易来实现。

每个以太坊的外部所有账户由一对密钥定义，一个是私钥，一个是公钥。区块链的 EOA 账户由它们的地址来做索引。取公钥的后 20 位作为地址，这和比特币的地址不一样。每个公私钥对被编码存放在一个密钥文件（Keyfile）中。密钥文件采用 JSON 格式，可以用文本编辑器打开来看。密钥文件的私钥都是用在建立账户时输入的口令来加密的。密钥文件存在以太坊节点的数据目录中的 keystore 子目录中。密钥文件需要经常备份，否则如果失掉密钥文件，账户里的以太币也就无法找回了。

合约账户可以执行图灵完备的计算任务，也可在合约账户之间传递消息，合约编译成以太坊虚拟机字节码（EthereumVirtual Machine Bytecode），并记录在区块链上。

外部所有账户可以通过发送交易到合约来实现对合同的调用。这需要提供几个参数，例如 EOA 的地址、合约的地址，以及数据。数据部分包括需要调用的合约里的方法（method）以及其传递的参数。这个需要用到 Application Binary Interface（ABI）来作为传递数据的编码和解码的标准。关于 ABI 的详细信息可以参考以太坊 wiki 网页 https://github.com/ethereum/wiki/wiki/Ethereum-Contract-ABI。

2. 区块链设计

比特币采用 Merkle 树来将交易的哈希值按一定算法组成二叉树状结构[6]，顶层节点的哈希值相当于整个交易清单的指纹，可以用来校验交易清单。中本聪采用 Merkle 树设计，也是为了轻量级节点能通过 SPV（简化支付验证）方式来方便地校验交易。SPV 不用下载整个交易清单，而是只需要区块报文头中交易清单顶层节点的哈希值，以及与自身节点相关的交易，然后可以通过向其他节点查询其他相邻交易，就可以完成对某个交易是否包含在区块链中某个区块的验证。

以太坊的区块链的每个区块不但保存着交易清单，还保存最新的状态。以太坊作为一个通用的区块链编程平台，引入了账户概念，由此它也带来更为复杂的校验和查询需求。例如要查询账户的余额或判断一个账户是否存在，光用比特币的 Merkle 树就满足不了要求。因此以太坊采用 Merkle Patricia 树来实现对交易和状态的校验和查询[7]。下

面看看交易和状态面临的问题。

以太坊的状态包含一个键值表（key-value map），其中键是地址，值是账户里声明的变量，包括余额、随机数（nonce）、代码和账户的存储（存储也以一棵树的形式来组织）。与交易数据只能增不能改不一样，账户的状态经常被改变，其余额、随机数经常变。

另外，新的账户也经常被插入，键在存储里也被经常插入和删除。因此 Merkle 树不适合这种情况，而需要一种可以在插入、更新和删除操作后快速计算新的树根哈希值，而不需要重新计算整棵树的数据结构。同时，树的深度是有限的，即使在有攻击者试图通过故意发很多交易来尽量增加树的深度的情况下，不然一个攻击者可以通过操纵树的深度，以使得每个更新都变得非常慢，来对平台实施拒绝服务攻击。

还有一个要求是树的根哈希只是与树的数据有关，而与更新的顺序无关。不同的更新顺序或者甚至重新计算整个树的根哈希值都不会改变树的根哈希值。Patricia 树是符合这些要求的数据结构。

简单来说，以太坊的账户的状态由键值表（Key-Value Map）来表示，在 Patricia 树里，键被编码成向下访问树的路径。在以太坊的 Patricia 里，每个节点有 16 个子节点，所以路径用十六进制来编码。例如，键“ dog”的十六进制编码是 6 4 6 15 6 7，所以要访问键“ dog”所对应的值，就必须先从根节点开始，向下到第 6 个节点，然后再由该节点向下到第 4 个节点，以此类推，一直到最后。

在区块链的区块报文头中，不像比特币那样仅仅存放一个交易清单的 Merkle 树根哈希值，而是存放了 3 个根哈希值：一个是交易的 Merkle 根哈希值，另外一个是状态的根哈希值，还有一个是收据的根哈希值。

另外一个和比特币的不同是，以太坊的区块链中的每个区块保存区块链号和区块难度。

3. PoW 机制

以太坊的 PoW（工作量证明）算法叫 Ethash 算法（是一个经过修改的 Dagger-Hashimoto 算法），该算法主要寻找一个随机数作为输入，使得运算结果小于一个特定的难度门槛。PoW 机制的前提是，不存在比逐个试更好的找到该随机数的方法，同时验证结果必须非常方便且成本小。由于哈希运算的结果是均匀分布（Uniform Distribution）的，所以可以保证，通常找到该随机数的时间取决于难度门槛。这样的话可以通过控制难度来控制在网络上找到一个新区块的时间。以太坊是通过动态调控难度来达到平均每 15s 在全网中找到一个新区块的。每 15s 的“心跳”基本上是全网更新系统状态的节奏，并保证当攻击者的计算能力不超过全网的计算能力的一半时，攻击者无法改写交易记录

或进行分叉（以便进行双花交易）。这就是所谓的“51%”攻击。网络上的矿工的挖矿收入期望直接反映他们拥有的计算力，或者哈希速率在整个网络中的占比。

比特币的工作量证明机制依靠的仅仅是 CPU 计算难度问题，以太坊的 Ethash 工作量证明机制加入内存难度，使得它具有抵抗单凭哈希运算优化的 ASIC 挖矿机的属性。内存难度是通过在算法设计中要求选择由随机数和区块报文头决定的一部分固定资源，这些资源一般是几个 GB 的数据，叫“有向无环图”（DAG）。每 30000 个区块后需要有一个全新的 DAG。这相当于一个 125 小时的窗口，或 5.2 天，称为一个 epoch。这个图需要一段时间才能生成。因为 DAG 只和区块链深度相关，因此可以提前生成。如果没有现成的，以太坊的客户端需要等生成了 DAG 后才能产生新的区块。这样的话在每个 epoch 转换的时候，如果客户端不预先生成 DAG，网络就会出现大规模的延迟。当一个矿工节点第一次启动时，需要等 DAG 生成之后才能开始挖矿。

以太坊的 Go 语言实现程序 geth 和 C 语言实现的挖矿程序 ethminer 都实现自动 DAG 生成，并在 epoch 转换的过程中维护两个 DAG。以太坊的 Ethash 算法可以在较慢的 CPU 环境中进行哈希运算，但在挖矿节点上可以通过增加内存和带宽来提升挖矿速度。对内存的高要求使得大型矿的矿主没有太大的比较性超线性收益；对带宽的高要求使得用堆超快的计算单元来共享存储的方法并不能带来更好的收益。这样对矿池挖矿来说没有太多好处，因此以太坊从设计上希望避免出现像比特币那样的矿池算力集中化的问题。

类似比特币，以太坊的挖矿静态收益将随着时间推移而逐渐减少，目前静态收益是每挖到一个区块获得 5 以太币。未来矿工的收益将主要依靠发送交易的用户支付的“燃料”来获取收益。以太坊的矿工奖励制度比比特币复杂。很多参考资料没有给出具体的奖励数额，有些甚至是错误的，例如对挖到“叔区块”的矿工奖励的描述，有些资料认为是（7/8）×5 = 4.375 以太币，其实这是不准确的。笔者参考了计算奖金的源代码，具体的奖金机制描述如下。

每当一个矿工挖到一个区块，他将获得 5.0 以太币（Ether）的静态收益，同时获得在区块上的“燃料”（gas），价值取决于当前的“燃料”价格。另外矿工也获得一个将“叔区块”（uncle）包含进区块链的额外奖励，相当于每包含一个 uncle 区块将获得（1/32）×5 以太币的收益。而产生“叔区块”的矿工将按下面的公式获得奖励：

挖到“叔区块”矿工奖励 =（叔区块 ID + 8– 当前区块 ID）×5/8

例如，假设当前区块 ID 是 1600，叔区块 ID 是 1598，那么挖到叔区块 1598 的矿工将获得（6/8）×5，等于 3.75 以太币。如果叔区块的 ID 是 1599，那么挖到叔区块 1599 的矿工将获得（7/8）×5，也就是 4.375 以太币的奖励。

注意：所谓“叔区块”，是指符合难度条件，但区块里的交易不被确认的区块，或叫“废块”（Stale）。比如矿工 A 挖到一个符合难度规定的合规区块 a，而几乎同时矿工 B 也挖到符合标准的区块 b，但由于网络延迟，区块 b 没有被确认，成了废块，而 a 成了网络共识的区块，被包括在区块链中。由于以太坊产生区块的速度比比特币产生区块的速度要快很多，因此在网络繁忙的时候，相对于比特币系统更容易出现“废块”。在比特币系统中，生产废块的矿工只能自认倒霉，是没有奖励的。而在以太坊中，产生“叔区块”的矿工和将“叔区块”包括在区块链上的矿工都能得到奖励。这样产生废块的算力也被包括进来，有效地增强了安全性，使得攻击者不容易追上一个带“叔区块”的主链。同时通过给“叔区块”奖励，也避免出现像比特币那样计算力高度集中的矿池，因为矿池相对来说不像单个挖矿节点那样容易产生废块。严格说来，“叔区块”是在当前链接区块往前推最多 6 个的“祖先”废块，每个区块最多能链接两个“叔区块”。

以太坊采用一个与比特币不同的算法叫 GHOST（幽灵）来构建区块链。GHOST 的全称是 Greedy Heaviest Observed Subtree，中文直译是“贪婪最重观察子树”。严格来说，以太坊的区块链不是一个链条，而像一棵树，它包含前面提到的“叔区块”。

在比特币系统中，矿工按一定的优先级把未确认的交易打包到新发现的区块上。交易的优先级按交易额和链龄（指 UTXO 存在的时间）来决定。交易额越大、链龄越高，优先级就越高。交易费用是用户的输入和输出之差。如果输入和输出之差为零，随着交易发生的时间越来越久，其在交易池的优先级会逐渐升高。因此一般来说，即使付给矿工零交易费用的交易都有机会被矿工包含在区块链上。当然个别矿工可以有自己的规则，可以拒绝零交易费的交易。而在以太坊平台，不提供“燃料”的交易不会被执行，也不会被包含在区块链上。以太坊的交易费用按以下公式计算：

$$\text{Total cost} = \text{gasUsed} \times \text{gasPrice}$$

其中 gasUsed 是执行该交易所消耗的燃料，燃料的价格由用户和矿工决定，一般来说在用户建一个交易的时候，可以提一个燃料价格。在以太坊发布的第一版本 Frontier（前线）中，以太坊客户端的默认燃料价格是 0.05e12 wei，大约是一亿分之五个以太币。矿工一般不会接受低于普遍价的燃料价格。

4. 计算和图灵完备

以太坊作为通用的区块链平台，需要提供比比特币更强大的计算能力。前面说过，

从安全角度出发，比特币的设计专门选择一个不具图灵完备性的脚本引擎，目前能通行的比特币脚本指令也不多，但在虚拟货币的应用场景已经是绰绰有余了。而在以太坊上，一个和比特币非常大的不同点就是选择了图灵完备的计算环境——以太坊虚拟机（EVM）。这就意味着在 EVM 上可以做所有的能想得到计算，包括无限循环。EVM 指令包括一个 JUMP 的跳转指令，可让程序跳回前面的程序代码，也可以像条件判断语句那样做条件跳转，当满足一定条件时将程序跳转到另一个地方执行。另外，一个合约可以调用其他合约，这提供了潜在的递归调用的功能。这就很自然地会导致一个问题：一个搞破坏的用户能否通过强制矿工或全节点进入死循环而将他们基本关掉呢？这个其实也是一个“停机问题”，也就是说，通常没有任何办法去判定一个程序会否停机。以太坊怎么解决这个问题呢？它首先要求每个交易要给出最大的计算步骤，交易的发起人要提供 Gas 作为交易费以供矿工把交易加进区块。如果实际运行超过了该最大计算步骤，计算将被终止，而交易费会归挖到区块的矿工所有。因此以太坊采用经济的方法来保证以太坊平台的安全。

以太坊网络的每个节点都运行 EVM 并执行合约代码，因此以太坊就像一个并行运行的“世界电脑”，在所有的节点上同时进行账户的状态转换，并形成网络层面对所有账户状态的共识。虽然这种 P2P 的运行方式并不是最高效的，但却是最有安全保障的，可以说，这部“世界电脑”永不停机。

5. EVM 高级语言

比特币不提供高级语言的支持，以太坊则提供高级语言让用户编写智能合约。以太坊的高级语言最后会编译成在 EVM 中执行的 EVM 字节码（bytecode），部署在以太坊区块链上。以太坊提供 3 种编程语言：Solidity、Serpent 和 LLL。

- Solidity 类似 JavaScript 语言，是目前以太坊上最流行的智能合约编程语言。
- Serpent 类似 Python 编程语言，它结合了低级语言的效率和易用的编程方式。Serpent 用 LLL 语言来编译。
- LLL 是 Lisp Like Language 的简称，顾名思义是一个像 Lisp 的语言。它有些像汇编语言，设计得非常简约，基本上就是在 EVM 上的一个微小的封装。

另外一个类似 C 的语言 Mutan 已经基本弃用，不再被维护。

6. 以太坊 P2P 网络

（1）RLPx 协议

以太坊网络节点间的通信采用 ÐΞVp2p 线上协议。节点间采用 RPLx[⊖]编码及认证

⊖ RLPx 是一个采用密码方式通信的 P2P 网络协议，为应用在 P2P 网络上通信提供通用的传输和接口。RLPx 是专门为去中心化应用设计的 P2P 通信协议。

的通信传输协议来传输消息包，即提供发送和接收消息的协议功能。节点可以自由地在任何 TCP 端口发布和接受连接，默认的端口是 30303。目前正式版的 RLPx 实现了以下功能：

- 单一协议的 UDP 节点发现
- ECDSA 签名的 UDP
- 加密握手 / 认证
- 节点持久性
- 加密 / 认证 TCP
- TCP 帧处理

ÐΞVp2p 节点采用 RLPx 的发现协议 DHT（Distributed Hash Table）来实现邻节点的发现。节点间的连接也可以通过具体客户端的 RPC API 进行，并提供对方端点来连接邻节点。

当两个节点连接并握手时，它们互相交换状态信息，状态信息包括总难度和它们的区块哈希值。总难度相当于节点所有的区块链上区块难度的总和。其中一个总难度小的节点会向对方索取对方整个区块链中区块的哈希值。这些哈希值的链条会存放在一个“工作池”中，供所有连接的邻节点共享。当一个节点发现在哈希链上有它没有的区块哈希值时，它将向邻节点索取从该哈希值所代表的区块起往后的 N 个区块，并做好标记，这样不会从另外一个节点获取这 N 个区块。

RLP（递归长度前缀）是一种编码方式，其目的是将二进制数据进行任意嵌套的数组编码。在以太坊里，RLP 是用来对对象进行编码的主要方法。

（2）Whisper 协议

Whisper 协议是 DApp 间通信的通信协议。Whisper 结合了 DHT 和数据包消息系统（如 UDP），因此同时具有以上两种协议的特性。Whisper 是一个纯标志（identify）的消息系统，它提供一个低层次的（非应用相关）但又简易使用的 API，而不需要记忆底层的硬件属性。另外，就像 DHT，有一个每条可配置的 TTL（Time to Live 生存时间）以及用来签名或加密的规则。在这个意义上，Whisper 提供多索引，非单一的记录，也就是说同一记录可以有多个键，有些键可能和别的记录一样。

Whispher 不是一个典型的通信系统，它并不是设计来取代 TCP/IP、UDP、HTTP 或其他常见的协议的，也不是用来提供一个面向连接的通信系统，更不是简单地在两个节点间传输数据。它的主要目标也不是提升带宽或降低延迟，而主要是直接设计来给新型的应用开发模式用的一个新的通信协议。它是从头设计的为简便而高效使用多播（multi-

casting）和广播（broadcasting）场景的协议。类似的低层次部分异步通信也是一个重要的设计目标。降低低价值流量或迟滞也是另一目标，这相当于 QoS 控制。Whisper 是为需要大规模的多对多数据发现、信号谈判和最少的传输通信、完全的隐私保护的下一代 DApp 而设计的。

Whisper 的使用场景有以下几种：

- DApp 需要把少量的信息发布出去，而这些发布的信息要保留相当一段时间。例如一个外汇交易所将一个货币的挂牌卖价发布出去。这个卖价可能需要保留几分钟或几天时间。
- DApp 需要发信号给其他 DApp，希望它们参与对某个交易的协同。
- DApp 之间需要提供非实时的提示或通常的通信，例如聊天室应用等。
- DApp 需要提供暗通信，也就是通信的双方除了知道对方的哈希值外，不知道对方更多的底细。

7. 事件

合约是在区块验证的时候被交易触发。如果设想在函数调用情况下，合约的执行是异步的，因此它没有返回的值。合约与外部的通信是通过日志事件实现的。日志事件是交易执行时产生的收据一部分。收据保存在收据树上，它的完整性由当前收据树的根保证，收据树的根和状态树以及交易树的根一起保存在区块的报文头。从外部的大的范围说，收据是以太坊系统状态的一部分，除了在合约内不能读取收据的数据以外。

以太坊中的事件是一个以太坊日志和事件监测的协议的抽象。日志的记录中提供合约的地址，一组最多 4 个议题（Topics）和一些任意长度的二进制数据。事件利用现有的 ABI 功能来解析日志记录。根据一个事件名和一些列的事件参数，可以把它们分为两个系列：一个是建立了索引的，一个是没有索引的。那些建了索引的（可以最多有 3 个）是用来和事件的 Keccak 哈希签名一起作为议题的日志记录；那些没有建立索引的用来组成事件的字节数组。

3.4　区块链 3.0 架构：超越货币、金融范围的区块链应用

《区块链：新经济蓝图及导读》一书的作者 Melanie Swan 把超越货币、金融范围的区块链应用归为区块链 3.0，特别是在政府、健康、科学、工业、文化和艺术领域的应用。它支持广义资产、广义交换，支持行业应用。

支持行业应用意味着区块链平台必须具备企业级属性。具体说来就是安全性的考

虑会更为突出，在很多企业级应用场景需要有授权才能访问区块链，也就是权限控制链（Permissioned Chain），一般来说企业级的区块链的部署模式是联盟链或私有链。另外区块链 3.0 也需要图灵完备的智能合约平台，同时对网络和共识算法的性能、每秒交易数（TPS）都有比较高的要求。因此，区块链 3.0 架构是分布式架构，但可以不是完全去中心化的架构，最有可能是在不同场景下的混合架构，也就是在部分场景，特别需要消费者参与共识过程的环境下可能用去中心化架构；而在很多企业使用场景，可能半中心化（例如联盟链场景）或部分中心化（私有链场景）更合适。

目前业界还没有一个成熟的区块链 3.0 平台。如图 3-10 所示是笔者根据区块链 3.0 应用的一些通用需求而设计的一个初步的通用架构。

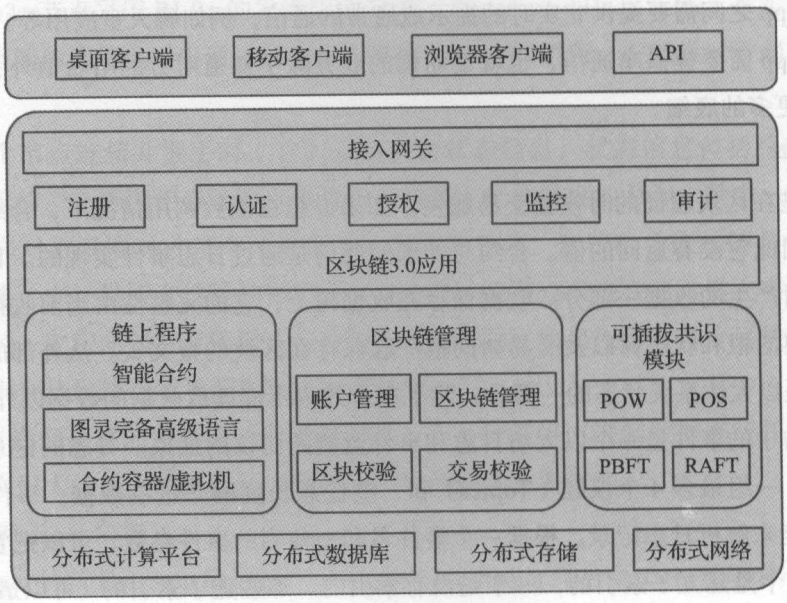


图 3-10　区块链 3.0 通用架构

目前，虽然像 NameCoin、Factom 等应用已属于区块链 3.0 的应用，但成熟的支撑区块链 3.0 应用的平台还不多见。HyperLedger 中的 Fabric 项目是基于 IBM 开源的 Open Blockchain 平台。Open Blockchain 平台定位于企业市场，是一个平台化设计，支持插件式共识算法的更换，以智能合约设计为中心，主要支持商用场景，其设计已接近区块链 3.0 的架构。参见第 8 章对 IBM Open Blockchain 架构的描述。

区块链 3.0 的应用场景很多，下面简单介绍一些典型的应用。

（1）自动化采购

采购方希望订立一个自动化的供货流程，追踪合约执行过程，并根据条件（时间、

地点、质量、数量）自动完成全额支付、部分支付、补贴、罚款。在此过程中会涉及多个采购方、供货方、物流、银行等，需要对每一批次商品的供货过程有完整记录。传统的解决方案存在的问题是：如何让各方遵守规则，完全按供货记录计算盈收？现在可以通过采用区块链方案，实现多方共同记账，共同监管，实现效率和透明度的提供，以及提高抗风险能力。

（2）智能化物联网应用

未来智能设备能够通过智能物联网代替人处理一些日常工作。例如汽车可以自动订购汽油、预定检修服务或清洗服务。冰箱可以自动订购食品，甚至空调和冰箱可以谈判商量如何错峰用电等。这里面碰到的问题就是智能设备是否可信，如何监控、管理分散的智能设备。采用区块链的方案，可以在一个分布式的物联网建立信用机制，利用区块链的记录来监控、管理智能设备，同时利用智能合约来规范智能设备的行为。

（3）供应链自动化管理

客户希望知道购买的商品的供应链信息，例如消费者希望知道食品的生产、加工、经销、仓储、运输过程，原材料的来源等；整机集成商希望知道部件的生产厂商、渠道来源等。现有的问题是商品供应链权属关系（原厂、总代、分销、零售）和上下游关系（生产、总装、维修、保养）都比较长，没有一个统一共享的数据平台，每个企业只知道一部分的信息。而且在生产过程中，商品形态也会出现很大的变化（如小麦、面粉、饼干），这使得商品的追踪溯源非常困难。而采用区块链的方案，可以登记每个商品的出处，提供一个共享的全局账本，追踪溯源所有引起状态变化的环境。这对生产过程、市场渠道的管理，以及政府监管都会有所帮助。

（4）虚拟资产兑换、转移

在游戏或某些行业中，消费者会积累很多虚拟资产（点数、积分、奖励、装备、战力）等。消费者希望能方便地将虚拟资产兑换或转移。比如游戏玩家希望游戏虚拟资产能从一个游戏转移到另一个游戏，或者玩家之间能够互相兑换这些虚拟资产。现在的问题是如何建立一个自动化的、可信任的交易平台来实现虚拟资产的安全、公正的转移。目前中心化的兑换平台很容易被运营商操纵。采用区块链的方案可以实现虚拟资产的公开、公正的转移，不受第三方影响，自动到账。

（5）产权登记

包括不动产、动产、知识产权、物权、租赁使用权益、商标、执照、许可、各类票据、证书、证明、身份、名称登记等在内的产权登记，都可以采用区块链技术来登记，以保证公正、防伪、不可篡改，以及可审计等。

未来，随着人工智能（AI）、物联网的发展，区块链会有越来越广泛的应用场景。在不久的将来，区块链将作为下一代互联网的重要组成部分，解决目前互联网存在的建立信用、维护信用成本高的问题，并将互联网从现在的信息互联网提升到价值互联网，这必将给人类社会的方方面面带来更大的革新，带来更大的正面影响。

3.5 互联链架构剖析

3.5.1 互联链背景

虽然作为虚拟货币的比特币非常成功，但金融行业更看重的是区块链技术在支付领域中的应用前景。现在的支付系统很多是竖井型，互不连接。如果在一个国家内，或者支付参与方账户同在一个支付网络或同一个账本中，支付还相对容易，如图 3-11 所示。但如果用户试图在不同账本体系间支付，就没有这么容易，如图 3-12 所示。比如说支付宝的用户想向微信支付里的账户转账，就没有这么容易。虽然账本之间存在连接，但这些连接都需要人工干预，交易的确认非常慢和昂贵。这里面碰到的首要问题就是一个信用的问题。也就是说，用户首先要信任在两个账本起连接作用的连接机构，相信它们不会卷款而逃。这些连接机构目前一般是大的银行，或支付组织，像 Visa、万事达、银联等。而这些大的机构往往要靠非常高的成本来建立信用。

图 3-11 同一账本下的支付转账

图 3-12 用户在不同账本体系之间的支付转账

比特币的出现，特别是其不依赖中心化机构而解决的双花问题，以及以极小的成本能使用户在陌生网络中建立信用，使得大家意识到传统的中心化支付系统需要重新构建。

从某种意义上来说，目前支付的状况有些类似在互联网出现之前的状况。互联网出现之前，不同系统之间没办法用标准的协议来通信、传递信息。互联网的出现解决了这个问题。互联网的架构，简单说来就是通过如图 3-13 所示的层次架构，在不同链路、不同物理层上的系统间实现应用层面的互联互通。

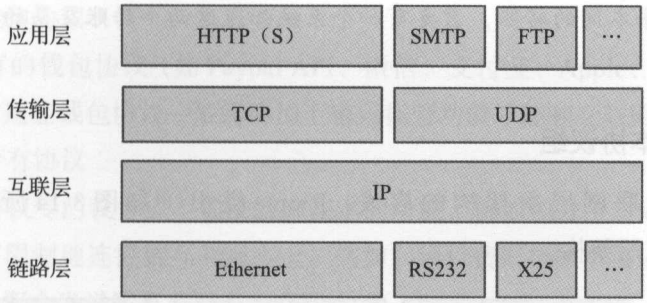


图 3-13　互联网层次架构

从区块链的发展来说，目前已经出现比特币、以太坊、NXT 等多种链，未来会出现更多不同的区块链。不同区块链之间的互联互通，需要由一个类似互联网概念的“互联链”的架构和协议来实现。我们先给出互联链的定义如下：

定义 7：互联链（Interchain）

提供实现不同区块链互联互通的统一架构和标准协议。

互联链概念还是一个全新的概念，目前还没有被业界普遍接受，因此相关的架构设计和标准化工作也没有开始。与互联链相关的一个类似概念——“互联账本”（Interledger）则由初创公司 Ripple Labs 提出。从某种意义上说，互联账本包含了互联链。也就是说，互联账本不单包含了在以区块链为基础的账本之间的互联互通，也包含了多种账本（传统账本、区块链账本）之间的互联互通。下面我们先介绍互联账本的概念。

3.5.2　互联账本

2012 年，分布式账本初创公司的 Ripple Labs 提出了一种与比特币不同的共识机制的分布式账本，并创造自己代币瑞波币（XRP）。2015 年，Ripple 提出一个互联账本（InterLedger）的协议，其目标是提供不同账本间的转账的通用协议，无论这些账本是分布式的，还是那些传统的中心化账本。如果用于基于区块链的账本，那就是提供互联链的一个参考架构模型和协议。

首先我们来看一下 Ripple 对账本系统的定义。

定义 8：账本（Ledger）

资金转账系统需要记录在保持状态的系统中，以防止同一笔资金被双花。这种保持状态的系统为账本（Ledger）。账本包含账户（Account），账户有资产余额，余额可以为正为负。

定义 9：连接器（Connectors）

资产可以在同一账本中的账户间转账，也可以在两个不同账本中的账户间转账。如

果在两个不同的账本间的转账，需要有一个系统知道这两个转账交易的关系，这个系统就是连接器。

3.5.3 互联账本协议组

受图 3-13 互联网层次架构的启发，Ripple 提出了如图 3-14 所示的互联账本（Interledger）协议层架构。

图 3-14 互联账本协议层架构

整个协议层分成 4 层。下面简单介绍这 4 层的功能。

1. 账本层

为了支持账户间的转账，账本必须实现一些 API 和协议，称为账本层（Ledger Layer）。这一层会有很多不同的协议，每种账本会对应一种协议，这包括专门设计的支持跨账本支付的简单账本协议（SLP），区块链中的比特币、以太坊等协议，以及传统的支付协议，像 ACH、ISO 20022、Paypal 等协议。

（1）简单账本协议

简单账本协议（Simple Ledger Protocol，SLP）是专门开发来提供一个跨账本支持的最基本功能的一个 RESTful，基于 JSONS 的协议。

（2）区块链协议（如比特币）

区块链是对一个单一共享状态提供共识的分布式的点对点（Peer-to-peer）系统。任何支持托管资金转账的区块链系统理论上来说都可以作为一个账本与互联账本连接。例如比特币支持多贷方和借方，以及 SHA256 哈希锁资金托管，这意味着它能进行 OTP/ILP 和 UTP/ILP 的互联账本交易。

（3）传统支付协议（如 ACH、ISO 20022）

传统协议一般不提供托管功能。在这种情况下，这些协议要就是用作协议升级，或者是用一个具高信用的机构，例如银行，作为托管提供方。

（4）专有钱包协议

有很多专有的钱包协议（如 Paypal API、微信、支付宝、Apple、Google），包括网页版和移动版，这些钱包协议一般需要加上密码托管功能才能和互联账本对接。

（5）其他专有协议

有些专有协议专门设计成不提供通用账本的功能，例如积分卡、点数和预付费账户等。这些可以有限制地连接到互联账本上。例如，预付费账户账本可以作为资金接收账本，但不能作为资金发送账本或资金中介账本。

2. 互联账本层

互联账协议（Interledger Protocol，ILP）保证不同的连接器能互操作，共同完成交易的路由。所有的互联账本传输都使用 ILP 和连接器通信，发送有关传输的请求。这包括报价请求和在另一个账本转账的需求。互联账本层协议定义一个使用账本、账户和金额的标准。这个用于交易路由以及使得报价有统一的口径。

（1）互联账本协议

用户当发起一个互联账本协议，发送方要用本地的账本层协议把资金发送到一个连接器。在这个过程中，发送方要发送一个告诉接收的连接器最终地址、转账资金额，以及转账条件的 ILP 包。一般来说，还要附加一个备忘信息。具体的传输方法取决于本地账本协议。

（2）互联账报价协议（Interledger Quoing Protocol，ILQP）

在互联网账本转账发生前，发送方要发一个报价请求给连接在同一账本的连接器，这个报价请求是通过互联账本报价协议发送的。发送方可以缓存报价请求，并多次发送给同一连接器。

（3）互联账本控制协议

互联账本控制协议（Interledger Control Protocol，ILCP）是一个连接器，用来交换路由信息和报告支付错误信息的协议。

3. 传输层

传输层（Transport Layer）协议负责协调不同的转账交易，使它们能进行跨账本转账的交易。转账的安全性取决于采用不同的传输协议。传输层为应用提供端到端的传输服务。目前有 3 种主要传输层协议：

- 乐观传输协议（OTP）
- 通用传输协议（UTP）
- 原子传输协议（ATP）

（1）乐观传输协议

乐观传输协议（Optimistic Transport Protocol，OTP）是一个没有托管的轻量级的传输协议。它只是简单地把资金传送给连接器。连接器可能会，也可能不会继续向对方账本发起转账请求，但无论如何，资金已离开资金发送方的账户。大多数情况下，OTP 是不适合的，但在一些重复多次发生的微小金额交易，资金丢失损失也不是很大的情况下，用 OTP 也是合理的。因此它只适合于微支付场景，这些场景下效率会比稳定性更重要。

（2）通用传输协议

通用传输协议（Universal Transport Protocol，UTP）是一个推荐采用的标准传输协议。它建立级联式托管传输来保障资金交付。UTP 用一个有托管的转账来保证在交易成功完成之前，资金不离开发送者的账户。它适合于任何金额的转账，应该作为缺省选择。

资金的托管必须支持采用密码学手段的托管。有托管的交易意味着有 4 种状态。

- 提议状态：资金没有发生转移。
- 准备状态：资金在托管状态。
- 执行状态：资金转账完成。
- 拒绝状态：资金转账被取消（资金返回给发送方）。

托管相当于金融领域里的“两阶段提交”（two-phase commit）。

（3）原子传输协议

原子传输协议（Atomic Transport Protocol，ATP）是最保守的协议，也是最复杂的互联账本协议。除了账本和连接器，它还包括一组独立的，被交易过程中所涉及的发送方、接收方和连接器认可的公证机构。由于每个参与方选择他们相信的一组公证机构，或许没有共同认可的工作机构，那 ATP 在这种情况下就不能使用。一般来说，可以基于一个所有参与方遵从的协议，采用一个自动化的流程来选择公证机构，或预先选好一组大家公认的公证机构。

ATP 用托管转账和一组预先协定的、可信任的公证来保证在多个账本中原子交易的完整性。这种协议的设立比较昂贵，但可以用来减少对对账的需求。

ATP 可以用在 UTP 交易中的一个子协议。当对方连接器给出一个比 UTP 传输更好的 ATP 传输报价时，连接器可采用建立一个 ATP 传输作为下一个段的传输。但这存在不能传输收据和公证机构故障的风险。

同样，在对方连接器不支持 ATP 传输的情况下，连接器可以决定用 UTP 作为下一段传输，这样一个 ATP 传输可以变成一个 UTP 传输。这也存在不能把收据发送给公证机构的风险。

4. 应用层

应用层（Application Layer）是互联账本协议的最高一层。应用协议层包括开放 Web 支付机制（OWPS）、简单支付建立协议（SPSP），和私有协议，如瑞波支付服务协议（RPSP）开放协议。这一层的协议负责谈判支付的以下关键属性：

- 源账户
- 目标账户
- 金额
- 条件

当这些属性定下来之后，应用层协议就会调用传输层协议发起支付。

（1）简单支付建立协议

简单支付建立协议（Simple Payment Setup Protocol，SPSP）是一个应用层协议，用于支付细节的谈判。SPSP 处理账户和金额发现、条件建立、报价和设置。SPSP 采用 Webfinger 和基于 HTTP 的协议来查询账户和金额细节。它采用 ILQP 做报价，采用 UTP 执行支付。

（2）定义其他应用层协议

其他方面的应用层协议应考虑以下问题：

- 账户发现
- 沟通金额和支付条件
- 在备忘信息中里的更多的详细信息
- 支持的或必须具备的条件类型
- 传输协议
- 接收支付交易的验证（金额、条件等）

通过互联账本协议，用户可在不同账本、不同支付网络中实现资金自动转账，交易的确认和验证时间可以大大缩短，而且更重要的是不必担心有资金丢失的风险。

3.5.4　互联账本各层协议关系

互联账本架构中，各层关系类似互联网的各层协议，每层只与相邻层做交互，不会跨层通信。支付由应用层发起，应用层调用传输层接口，传输层再调用互联账本层的接口，最后互联账本层通过调用账本层接口把包含目的地址连接器的支付请求发到账本层的连接器，账本层的连接器根据目的地址将支付请求发到对方账本的连接器，实现支付请求的发起。目的账本的连接器在接收到支付请求后，通过调用上层的互联账本层接口，把请求转到传输层，进而再转发到应用层，这样接收方接收到请求。支付的响应也

通过相同的路径返回给支付发起方，实现支付功能，如图 3-15 所示。

图 3-15 互联账本自动安全转账

互联账本协议还处于非常初级的概念模型阶段，离实际应用还有很大的一段距离。相信在互联账本方面的实践经验积累，会为未来互联链的架构和标准协议的制定提供一个参考。

3.6 本章小结

作为一名对区块链有所了解的技术人员，如何掌握区块链的架构呢？由于目前深度介绍区块链技术架构的书籍不多，这是困扰很多区块链技术爱好者的问题。本章深度剖析区块链 1.0、区块链 2.0 架构，同时也介绍了具有前瞻性的区块链 3.0 架构。学习本章，可以深入了解在组件级的比特币架构，同时了解以以太坊为代表的区块链 2.0 架构和比特币架构的异同。帮助读者在实际应用场景中选择合适的区块链架构以确保区块链应用的成功落地。最后介绍了互联链架构，使用户了解集成、整合不同账本的解决方案基础。

参考资料

[1] Coulouris, George; Jean Dollimore; Tim Kindberg; Gordon Blair, Distributed Systems: Concepts and Design (5th Edition). Boston: Addison-Wesley, 2011.

[2] 吴玉征，郑凯，中国云力量，电子工业出版社，2016.

[3] Stefan Thomas & Evan Schwartz, A Protocol for Interledger Payments, Ripple.com, 2015.

[4] Arvind Narayanan, Joseph Bonneau, Edward Felten, Andrew Miller, Steven Goldfeder, Bitcoin and Cryptocurrency Technologies, Princeton University Press, 2016.

[5] Mastering Bitcoin, O'Reilly Media, Inc.

[6] Vitalik Buterin.Merkling inEthereum. https://blog.ethereum.org/2015/11/15/merkling-in-ethereum. November 15th, 2015.

[7] Ethereum Wiki, Patricia Tree. https://github.com/ethereum/wiki/wiki/Patricia-Tree.

第4章 区块链中的密码学技术

为保证存储于区块链中的信息的安全与完整，区块及区块链的定义和构造中使用了包含密码哈希函数和椭圆曲线公钥密码技术在内的大量的现代密码学技术，同时，这些密码学技术也被用于设计基于工作量证明的共识算法并识别用户。

4.1 哈希算法

密码哈希函数是一类数学函数，可以在有限合理的时间内，将任意长度的消息压缩为固定长度的二进制串，其输出值称为哈希值，也称为散列值。以哈希函数为基础构造的哈希算法，在现代密码学中扮演着重要的角色，常用于实现数据完整性和实体认证，同时也构成多种密码体制和协议的安全保障。

碰撞是与哈希函数相关的重要概念，体现着哈希函数的安全性，所谓碰撞是指两个不同的消息在同一个哈希函数作用下，具有相同的哈希值。哈希函数的安全性是指在现有的计算资源（包括时间、空间、资金等）下，找到一个碰撞是不可行的。

在比特币系统中使用了两个密码学哈希函数，一个是 SHA256，另一个是 RIPEMD160。RIPEMD160 主要用于生成比特币地址，我们着重分析比特币中用得最多的 SHA256 算法。

SHA256 属于著名的 SHA 家族一员。SHA（Secure Hash Algorithm，安全哈希算法）是一类由美国国家标准与技术研究院（NIST）发布的密码哈希函数。正式名称为 SHA 的第一个成员发布于 1993 年，两年之后，著名的 SHA-1 发布，之后另外的 4 种变

体相继发布，包括 SHA224、SHA256、SHA384 和 SHA512，这些算法也被称作 SHA2。SHA256 算法是 SHA2 算法簇中的一类。对于长度小于 2^{64} 位的消息，SHA256 会产生一个 256 位的消息摘要。SHA256 具有密码哈希函数的一般特性。

SHA256 是构造区块链所用的主要密码哈希函数。无论是区块的头部信息还是交易数据，都使用这个哈希函数去计算相关数据的哈希值，以保证数据的完整性。同时，在比特币系统中，基于寻找给定前缀的 SHA256 哈希值，设计了工作量证明的共识机制；SHA256 也被用于构造比特币地址，即用来识别不同的用户。

SHA256 是一个 Merkle-Damgard 结构的迭代哈希函数，其计算过程分为两个阶段：消息的预处理和主循环。在消息的预处理阶段，主要完成消息的填充和扩展填充，将所输入的原始消息转化为 n 个 512 比特的消息块，之后对每个消息块利用 SHA256 压缩函数进行处理，SHA256 的计算流程如图 4-1 所示。这个计算流程是一个迭代计算的过程，当最后 1 个消息块（第 n 块）处理完毕以后，最终的输出值就是所输入的原始消息的 SHA256 值。

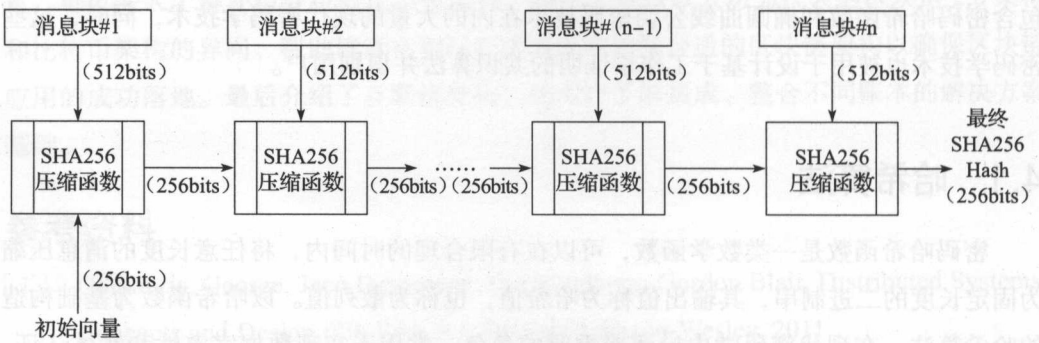


图 4-1　SHA256 计算流程

在比特币系统中，SHA256 算法的一个主要用途是完成 PoW（工作量证明）计算。按照比特币的设计初衷，PoW 要求钱包（节点）数和算力值大致匹配，因为需要通过 CPU 的计算能力来进行投票。然而随着人们对 SHA256 的计算由 CPU 逐渐升级到 GPU，到 FPGA，直至到 ASIC 矿机，这使得节点数和 PoW 算力也渐渐失配。解决这个问题的一个思路是引入另外的一些哈希函数来实现 PoW。

scrypt 算法最早用于基于口令的密钥生成，该算法进行多次带参数的 SHA256 计算，即基于 SHA256 的消息认证码计算，这类计算需要大量的内存支持。采用 scrypt 算法进行 PoW 计算，将 PoW 计算由已有的拼算力在一定程度上转化为拼内存，能够使得节点数和 PoW 的计算力的失配现象得到缓解。莱特币就是采用 scrypt 算法完成 PoW 计

算的。

SHA3 算法是 2012 年 10 月由 NIST 所选定的下一代密码哈希算法。在遴选 SHA3 算法过程中人们提出了一系列的候选算法，包括了 BLAKE、Grostl、JH、Keccak、Skein、ECHO、Luffa、BMW、CubeHash、SHAvite、SMID 等，最后胜出的是 Keccak 算法。达世币（DASH，原名暗黑币，DarkCoin）定义了顺序调用上述 11 个哈希算法的 X11 算法，并利用这个算法完成 PoW 计算。同样，由于采用了 X11 算法，使得节点数和 PoW 的计算力能够保持一定程度上的匹配。

4.1.1　哈希函数的性质与应用

我们来看看密码学哈希函数的 3 个主要性质及其应用。

1. 抗碰撞性

如前所述，碰撞是与哈希函数相关的重要概念，所谓碰撞是指两个不同的消息在同一个哈希函数作用下，具有相同的哈希值。哈希函数 H 的抗碰撞性是指寻找两个能够产生碰撞的消息在计算上是不可行的。值得注意的是找到两个碰撞的消息在计算上不可行，并不意味着不存在两个碰撞的消息。由于哈希函数把大空间上的消息压缩到小空间上，碰撞肯定存在。例如，如果哈希值的长度固定为 256 位，显然如果顺序取 1，2，…，$2^{256}+1$ 这 $2^{256}+1$ 个输入值，逐一计算其哈希值，肯定能够找到两个输入值，使得它们的哈希值相同。

一般，依据生日悖论，如果随机挑选其中的 $2^{130}+1$ 个输入，则有 99.8% 的概率可以发现至少一对碰撞的输入。然而，这样的计算非常耗时以至于计算不可行。对于哈希值长度为 256 位的哈希函数，要找到碰撞对，平均需要完成 2^{128} 次哈希计算，如果计算机每秒能够进行 10 000 次哈希计算，则需要约 10^{27} 年才能完成这 2^{128} 次哈希计算。

哈希函数的抗碰撞（collition-resistance）特性常被用来进行完整性验证。完整性是信息安全的 3 个基本要素之一，是指传输、存储信息的过程中，信息不被未授权的篡改或篡改后能被及时发现。由于哈希函数的抗碰撞性，我们可以把哈希值作为原输入消息的指纹（因为很难找到另一个消息经哈希运算之后得到相同的哈希值）。如果原消息在传输过程中被篡改，那么运行哈希函数后得到的新哈希值就会和原来的哈希值不一样，这样很容易就能发现消息在传输过程中完整性受损。对区块链来说，哈希函数的抗碰撞性可以用来做区块和交易的完整性交易。在区块链中，某个区块的头部信息中会存储着前一个区块的信息的哈希值，如果能拿到前一个区块的信息，任何用户都可以比对计算出来的哈希值和存储的哈希值，来检测前一个区块的信息的完整性。

2. 原像不可逆

原像不可逆通俗地说，指的是知道输入值，很容易通过哈希函数计算出哈希值；但知道哈希值，没有办法计算出原来的输入值。哈希函数的原像是不可逆的，这意味着依据哈希函数的输出是不能计算出该哈希函数的输入的，即已知 H(m)，试图计算出原始的 m 的值在计算上是不可行的。更特别地，若对消息 m 进行哈希计算时，引入一个随机的前缀 r，依据哈希值 H($r \| m$)，难以恢复出消息 m，这代表着该哈希函数值隐藏了消息 m。

承诺方案（Commitment Scheme）被认为是密码学领域中一类重要的密码学基本模型，承诺（Commitment）具有隐藏性和保密性。承诺模型可以看作一个密封信件的数字等价体。如果 Alice 想承诺某个信息 m，则她可把 m 放入一个密封的信封内，而无论什么时候她想公开这个信息，则只需要打开信封。这个过程要求数字信件能够隐藏信息，即承诺的隐藏性，同时 Alice 也不能改变 m；而通过承诺的打开，任何人都能验证他所得到的 m 其实就是 Alice 最初承诺的信息 m，即承诺的绑定性。

承诺方案包含以下两个算法。

- 承诺值计算 commit(m, r)：输入消息 m 和随机值 r，返回承诺值 c(=commit(m, r))。
- 承诺验证 verify(c, m, r)：输入承诺 c，消息 m 和随机值 r，若 c==commit(m, r)，返回真，否则返回假。

显然，如果定义 commit(m, r) = H($r \| m$)，利用哈希函数 H 的抗碰撞性和原像不可逆，承诺的隐藏性和绑定性均能成立，可以实现承诺方案。

3. 难题友好性

通俗地说，难题友好性（Puzzle Friendliness）指的是没有便捷的方法去产生一满足特殊要求的哈希值。正式的定义是：一个哈希函数 H 称为难题友好的，如果对于每个 n 位的输出 y，若 k 是从一个具有较高不可预测性（高小熵，英文为 high min-entropy）分布中选取的，不可能以小于 2^n 的时间找到一个 x，使 H($k \| x$) = y。这意味着如果有人想通过锁定哈希函数来产生一些特殊的输出 y，而部分输入值以随机方式选定，则很难找到另外一个值，使得其哈希值正好等于 y。

考虑一个由哈希函数构成的解谜问题：已知哈希函数 H，一个高小熵分布的值 *value* 以及目标范围 Y，寻找 x，使得 H(*value* $\| x$) $\in Y$。

这个问题等价于需要找到一个输入值，使得输出值落在目标范围 Y 内，而 Y 往往是所有的输出值的一个子集。实际上，如果一个哈希函数 H 的输出为 n 位，那么输出值可以是任何一个 0~2^n 范围内的值。预定义的目标范围 Y 的大小决定了这个问题的求解难度。如果 Y 包含所有 n 比特的串，那么这个问题就简单平常了；但如果 Y 只包含一个元

素，那么这个求解是最难的，相当于给定一个哈希值，找出其中的一个原像。事实上，由于 *value* 具有高小熵分布，这确保了除了随机尝试 x 值以完成搜寻那个很大的空间外，没有其他有效的途径了。

> 提示：小熵（min-entropy）是信息理论中衡量某个结果的可预测性的一个指标。高小熵指的是变量呈均匀分布（随机分布）。如果我们从对分布的值进行随机抽样，不会经常抽到一个固定的值。例如，如果在一个 128 位的数中随机选一个固定的数 n，那么选到该数的几率是 $1/2^{128}$。

哈希函数的难题友好性构成了基于工作量证明的共识算法的基础。例如，给定字符串“blockchain”，并在这个字符串后面连接一个整数值串 x，对连接后的字符串进行 SHA256 哈希运算，要求得到的哈希结果（以十六进制的形式表示）以若干个 0 开头的。按照这个规则，由 x=1 出发，递增 x 的值，我们需要经过 2688 次哈希计算才能找到前 3 位均为 0 的哈希值，而要找到前 6 位均为 0 的哈希值，则需进行 620 969 次哈希计算。也就是说，没有更快捷的方法来产生一个满足要求的哈希结果。这样通过哈希运算得出的符合特定要求的哈希值，可以作为共识算法中的工作量证明。

4.1.2　哈希指针链

哈希指针是一类数据结构，除了包含通常的指针外，还包含一些数据信息以及与这些信息相关的密码哈希值，这就使得正常的指针可用于取回信息，哈希指针用于验证信息是否发生改变，图 4-2 表示了一个哈希指针。

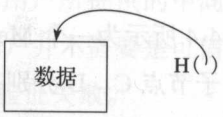


图 4-2　哈希指针

区块链就可以看作一类使用哈希指针的链表，如图 4-3 所示。这个链表链接一系列的区块，每个区块包含数据以及指向表中前一个区块的指针。区块链中，前一个区块指针由哈希指针所替换，因此每个区块不仅仅告诉前一个区块的位置，也提供一个哈希值去验证这个区块所包含的数据是否发生改变。

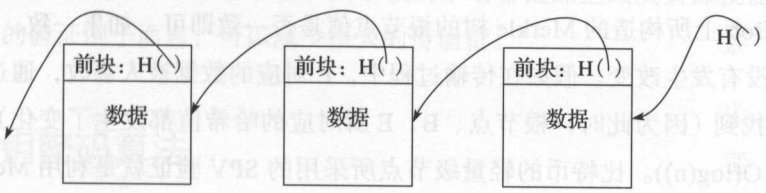


图 4-3　哈希指针链

我们可以使用区块链去构造一个防篡改的日志系统。在这个系统中，基于区块链的日志节点链表被用来存储数据，链表节点通过哈希指针链接，新节点追加在日志链表的尾部。同时，日志链表的头哈希指针所指向的头节点内容不可改变。若日志链表中的某个节点的数据被篡改，则系统能够检测出来。

不妨假定攻击者改变了节点 k 的数据，由于其后继节点 $k+1$ 存储了节点 k 的哈希值，由于密码哈希函数的抗碰撞性，通过简单地计算节点 k 的数据的哈希值，就能发现计算出的值与节点 $k+1$ 的哈希指针值不一致，于是可以断定节点 k 或节点 $k+1$ 的信息被篡改。当然，攻击者可能能够连续改变前一个节点的哈希值来掩盖不同，但这个策略在处理日志链表的头节点时将会失败。特别地，一旦我们将链表头部的哈希指针存储在不能改变的地方，攻击者将不能改变任何节点而不被发觉。

因此，若攻击者想在日志链表中的任意位置改变数据，为保持一致性，他必须向表头方向修改所有的哈希指针，最终由于不能改变链表头部而失败。因此，只需单个哈希指针，基本上就能保证整个链表的哈希值的一致性，从而达到防篡改的目的。

4.2 Merkle 树

Merkle 哈希树是一类基于哈希值的二叉树或多叉树，其叶子节点上的值通常为数据块的哈希值，而非叶子节点上的值是将该节点的所有子节点的组合结果的哈希值。如图 4-4 所示为一个 Merkle 哈希树，节点 A 的值必须通过节点 C、D 上的值计算而得到。叶子节点 C、D 分别存储数据块 001 和 002 的哈希值，而非叶子节点 A 存储的是其子节点 C、D 的组合的哈希值，这类非叶子节点的哈希值被称作路径哈希值，而叶子节点的哈希值是实际数据的哈希值。

在计算机领域，Merkle 树大多用来进行完整性验证处理。在处理完整性验证的应用场景中，特别是在分布式环境下进行这样的验证时，Merkle 树会大大减少数据的传输量以及计算的复杂度。例如，以图 4-4 为例，若 C、D、E 和 F 存储了一组数据块的哈希值，当把这些数据从 Alice 传输到 Bob 后，为验证传输到 Bob 的数据完整性，只需要验证 Alice 和 Bob 上所构造的 Merkle 树的根节点值是否一致即可。如果一致，表示数据在传输过程中没有发生改变。假如在传输过程中，E 对应的数据被人篡改，通过 Merkle 树很容易定位找到（因为此时，根节点、B、E 所对应的哈希值都发生了变化），定位的时间复杂度为 O(log(n))。比特币的轻量级节点所采用的 SPV 验证就是利用 Merkle 树这一优点。

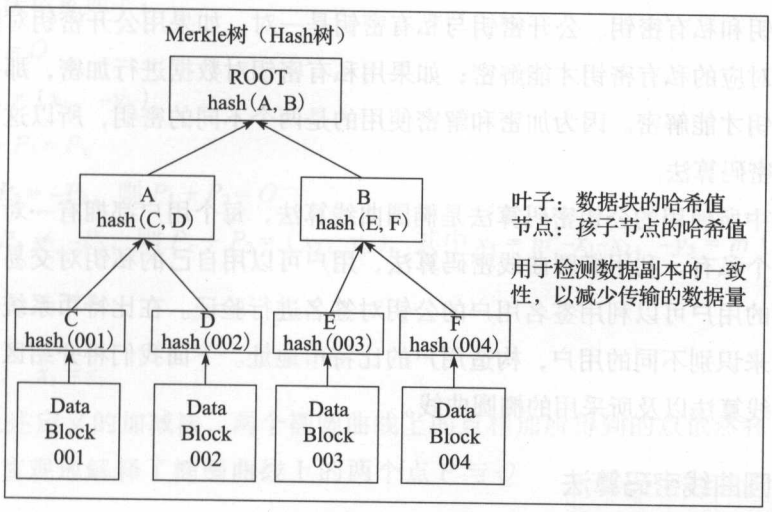


图 4-4　Merkle 哈希树

利用一个节点出发到达 Merkle 树的根所经过的路径上存储的哈希值，可以构造一个 Merkle 证明，验证范围可以是单个哈希值这样的少量数据，也可以是验证可能扩至无限规模的大量数据。

区块链中的 Merkle 树是二叉树，用于存储交易信息。每个交易两两配对，构成 Merkle 树的叶子节点，进而生成整个 Merkle 树。

Merkle 树使得用户可以通过从区块头得到的 Merkle 树根和别的用户所提供的中间哈希值列表去验证某个交易是否包含在区块中。提供中间哈希值的用户并不需要是可信的，因为伪造区块头的代价很高，而中间哈希值如果伪造的话会导致验证失败。

如图 4-4 所示，为验证数据块 003 所对应的交易包含在区块中，除了 Merkle 树根外，用户只需要节点 A 对应的哈希值 Hash(C, D) 以及节点 F 所对应的哈希值 Hash(004)。除了数据块 003 外，他并不需要其他数据块所对应的交易明细。通过 3 次哈希计算，用户就能够确认数据块 003 所对应的交易是否包含在区块中。实际上，若区块包含图 4-4 所对应的 Merkle 树，且区块所包含的 4 个交易的容量均达到最大值，下载整个区块可能需要超过 400 000 个字节，而下载两个哈希值加上区块头部仅需要 120 个字节。就我们的验证例子而言，可以减少很大的传输量。

4.3　公钥密码算法

公钥密码算法是现代密码学发展过程中的一个里程碑。这类密码算法需要两个密

钥：公开密钥和私有密钥。公开密钥与私有密钥是一对，如果用公开密钥对数据进行加密，只有用对应的私有密钥才能解密；如果用私有密钥对数据进行加密，那么只有用对应的公开密钥才能解密。因为加密和解密使用的是两个不同的密钥，所以这种算法也可叫作非对称密码算法。

区块链中所使用的公钥密码算法是椭圆曲线算法，每个用户都拥有一对密钥，一个公开，另一个私有。利用椭圆曲线密码算法，用户可以用自己的私钥对交易信息进行签名，同时别的用户可以利用签名用户的公钥对签名进行验证。在比特币系统中，用户的公钥也被用来识别不同的用户，构造用户的比特币地址。下面我们将介绍区块链中所涉及的椭圆曲线算法以及所采用的椭圆曲线。

4.3.1 椭圆曲线密码算法

椭圆曲线密码（Elliptic Curve Cryptography，ECC）算法是基于椭圆曲线数学的一种公钥密码的算法，其安全性依赖于椭圆曲线离散对数问题的困难性。

椭圆曲线密码算法具有下面两个明显的优点：

1）短的密钥长度，这意味着小的带宽和存储要求；

2）所有的用户可以选择同一基域上的不同的椭圆曲线，可使所有的用户使用同样的操作完成域运算。

椭圆曲线可以定义如下：

设 p 是一个大于 3 的素数，在有限域 F_p 上的椭圆曲线 $y^2 = x^3 + ax + b$ 由一个基于同余式 $y^2 = x^3 + ax + b \bmod p$ 的解集 $(x, y) \in F_p \times F_p$ 和一个称为无穷远点的特定点 O 组成，这里 $a, b \in F_p$ 是两个满足 $4a^3 + 27b^2 \neq 0 \bmod p$ 的常数。

图 4-5 显示了两种实际的椭圆曲线。

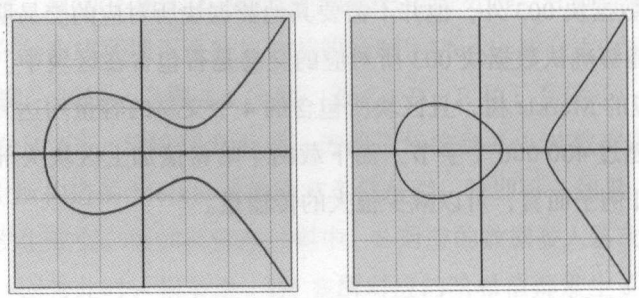


图 4-5 椭圆曲线

设 $P_1 = (x_1, y_1)$ 与 $P_2 = (x_2, y_2)$ 为椭圆曲线上的两个点，我们可以定义椭圆曲线上

的加法和减法运算如下：

1）$-O=O$

2）$-P_1=(x_1, -y_1)$

3）$O+P_1=P_1$

4）若 $P_2=-P_2$，则 $P_1+P_2=O$

5）若 $P_2 \neq -P_1$，则 $P_1+P_2=(x_3, y_3)$，其中 $x_3=m^2-x_1-x_2$，$-y_3=m(x_3-x_1)+y_1$，

$$m=\begin{cases}\dfrac{y_2-y_1}{x_2-x_1}, & x_2 \neq x_1 \\ \dfrac{3x_1^2+a}{2y_1}, & x_2=x_1\end{cases}$$

依据上述定义的加减法，两个椭圆曲线上的点相加所得到的点依然在原椭圆曲线上。图 4-6 直观地解释了椭圆曲线上的两个点 P 与 Q 相加的结果。

由此，在等式 $kP=P+P+\cdots+P=Q$ 中，已知 k 和点 P，求点 Q 比较容易，反之已知点 Q 和点 P，求 k 却是相当困难的，这个问题称为椭圆曲线上点群的离散对数问题。椭圆曲线密码体制正是利用这个困难问题设计的。在实际应用中，k 作为私有密钥，而 Q 作为公开密钥。

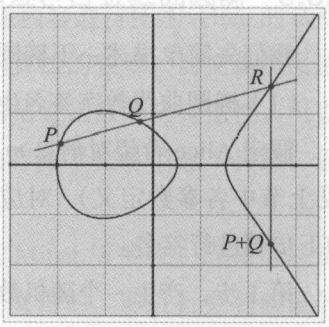


图 4-6　椭圆曲线上的两个点相加

4.3.2　secp256k1 椭圆曲线

比特币系统的区块链实现中使用的椭圆曲线为 Certicom 推荐的椭圆曲线 secp256k1。

Certicom 是国际上著名的椭圆曲线密码技术公司，已授权 300 多家企业使用 ECC 密码技术，secp256k1 为基于 F_p 有限域上的椭圆曲线，由于其构造的特殊性，其优化后的实现比其他曲线性能上可以提高 30%，对比 NIST 推荐的曲线，secp256k1 的常数以可以预测的方法选择，可以有效避免后门出现的可能性。

secp256k1 曲线形如 $y^2=x^3+ax+b$，由六元组 $D=(p, a, b, G, n, h)$ 定义，其中：

p = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFC2F

$=2^{256}-2^{32}-2^9-2^8-2^7-2^6-2^4-1$

a = 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

b = 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000007

压缩形式表示的基点 G 为：

G = 02 79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D9 59F2815B 16F81798

而非压缩形式的表示为：

G = 04 79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D9 59F2815B 16F81798 483ADA77 26A3C465 5DA4FBFC 0E1108A8 FD17B448 A6855419 9C47D08F FB10D4B8

G 的阶为：

n = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE BAAEDCE6 AF48A03B BFD25E8C D0364141

协因子为：

$$h = 01$$

4.3.3 椭圆曲线签名与验证签名

我们先简单描述一下椭圆曲线的签名与验证算法。

（1）椭圆曲线数字签名生成

假定 Alice 希望对消息 m 进行签名，她所采用的椭圆曲线参数为 D =(p, a, b, G, n, h)（见上节中各参数定义），对应的密钥对为 (k, Q)，其中 Q 为公钥，k 为私钥。Alice 将按如下步骤进行签名。

第 1 步，产生一个随机数 d，$1 \leqslant d \leqslant n-1$；

第 2 步，计算 $dG = (x_1, y_1)$，将 x_1 转化为整数 $\overline{x_1}$；

第 3 步，计算 $r = \overline{x_1} \bmod n$，若 $r = 0$，则转向第 1 步；

第 4 步，计算 $d^{-1} \bmod n$；

第 5 步，计算哈希值 H(m)，并将得到的比特串转化为整数 e；

第 6 步，计算 $s = d^{-1}(e + kr) \bmod n$，若 $s = 0$，则转向第一步；

第 7 步，(r, s) 即为 Alice 对消息 m 的签名。

（2）椭圆曲线签名验证

为验证 Alice 对消息 m 的签名 (r, s)，矿工（Miner）可以得到 Alice 所用的椭圆曲线参数以及 Alice 的公钥 Q。矿工将按以下步骤操作。

第 1 步，验证 r 和 s 是区间 [1, n−1] 上的整数；

第 2 步，计算 H(m) 并将其转化为整数 e；

第 3 步，计算 $w = s^{-1} \bmod n$；

第 4 步，计算 $u_1 = ew \bmod n$ 以及 $u_2 = rw \bmod n$；

第 5 步，计算 $X = u_1G + u_2Q$；

第 6 步，若 $X = O$，则拒绝签名，否则将 X 的 x 坐标 x_1 转化为整数 $\overline{x_1}$，并计算 $v = \overline{x_1} \bmod n$；

第 7 步，当且仅当 $v = r$ 时，签名通过验证。

为具体说明椭圆曲线签名和验证算法的过程，我们来看一个简化的例子：Alice 决定把 10 个比特币支付给 Bob，矿工负责把这笔账给记录下来。这个过程是怎么使用签名和验证算法进行的呢？

首先 Alice 从自己钱包中取出 10 个比特币，要将这 10 个比特币支付给 Bob，于是交易消息 m 产生了：Alice 支付 10 BTC 给 Bob。由于这个消息需要向全网广播，收到这个交易消息的用户会发生疑问：这个交易是不是真的？为打消其他用户的疑虑，Alice 需要对这段交易消息进行数字签名，以向大家确定这个交易确实是 Alice 发出的。为此，Alice 使用了 secp256k1 椭圆曲线。签名本质上是对交易消息内容进行使用 Alice 的私钥 k 加密（签名算法的第 6 步）。考虑到消息的规模和公钥密码算法的效率，对交易消息进行的签名实际上是对交易消息的哈希值进行签名，由于密码哈希函数的抗碰撞性，可以认为这样的转化是合理有效的。于是 Alice 向全网广播的内容除了交易消息本身外，还包含 Alice 对消息的签名以及 Alice 的公钥信息。

其次，Alice 发送的交易消息连同签名发出后，为矿工 Miner 所接收。为在区块链中记录这一交易，矿工首先需要验证这个交易是不是 Alice 发出的，即进行签名验证的工作。为此，Miner 也使用了同样的 secp256k1 椭圆曲线。对 Alice 签名验证的过程可以看作利用 Alice 公钥进行解密的过程，如签名验证算法中的第 5 步就使用了 Alice 的公钥 Q。当一切顺利的话，Miner 可以验证交易消息：Alice 支付 10 BTC 给 Bob 确实是 Alice 发出的，Miner 可以在之后的操作中把这个交易记入区块链中。如果签名验证失败，表明 Miner 收到的这个消息存在问题，Miner 会放弃将相关的交易记入区块链的操作。

利用椭圆曲线的签名和验证算法，一方面可以保证用户的账户不被冒名顶替，另一方面也能确保用户不能否认其所签名的交易。用户发起交易的时候，使用自己的私钥对交易信息签名，矿工收到信息后用用户的公钥对签名进行验证，一旦通过，该交易信息就可通过矿工进行记账，最终完成交易。

4.4　本章小结

本章简介了区块链中的密码技术。区块链通常并不直接保存原始数据或交易记录，

而是保存其哈希函数值，更具体的，比特币区块链通常采用双 SHA256 哈希函数，即将任意长度的原始数据经过两次 SHA256 哈希运算后转换为长度为 256 位（32 字节）的二进制数字来统一存储和识别。为快速归纳和校验区块数据的存在性和完整性，Merkle 树成为区块链的重要数据结构。公钥密码系统被用来实现区块链中的数据签名，比特币区块链中采用了椭圆曲线公钥密码系统，使用了 Certicom 推荐的 secp256k1 椭圆曲线。

参考资料

[1] Arvind Narayanan, Joseph Bonneau, Edward Felten, Andrew Miller, Steven Goldfeder. Bitcoin and Cryptocurrency Technologies. Princeton University Press, 2016.

[2] （加）Douglas R. Stinson. 密码学原理与实践（第 3 版）电子工业出版社 2009.

[3] https://en.wikipedia.org/wiki/SHA-3.

[4] https://en.wikipedia.org/wiki/Scrypt.

第5章

共识算法详解

在本书第 3 章我们提到，区块链架构是一种分布式的架构。其部署模式有公共链、联盟链、私有链三种，对应的是去中心化分布式系统、部分去中心化分布式系统和弱中心分布式系统。

分布式系统中，多个主机通过异步通信方式组成网络集群。在这样的一个异步系统中，需要主机之间进行状态复制，以保证每个主机达成一致的状态共识。然而，异步系统中，可能出现无法通信的故障主机，而主机的性能可能下降，网络可能拥塞，这些可能导致错误信息在系统内传播。因此需要在默认不可靠的异步网络中定义容错协议，以确保各主机达成安全可靠的状态共识。

利用区块链构造基于互联网的去中心化账本，需要解决的首要问题是如何实现不同账本节点上的账本数据的一致性和正确性。这就需要借鉴已有的在分布式系统中实现状态共识的算法，确定网络中选择记账节点的机制，以及如何保障账本数据在全网中形成正确、一致的共识。

在 20 世纪 80 年代出现的分布式系统共识算法，是区块链共识算法的基础。我们下面从基本的拜占庭容错技术入手，逐步介绍适合于私有链 / 联盟链和公共链的共识算法。

5.1 拜占庭容错技术

拜占庭容错技术（Byzantine Fault Tolerance，BFT）是一类分布式计算领域的容错技术。拜占庭假设是对现实世界的模型化，由于硬件错误、网络拥塞或中断以及遭到恶

意攻击等原因，计算机和网络可能出现不可预料的行为。拜占庭容错技术被设计用来处理这些异常行为，并满足所要解决的问题的规范要求。

5.1.1 拜占庭将军问题

拜占庭容错技术来源于拜占庭将军问题。拜占庭将军问题是 Leslie Lamport 在 20 世纪 80 年代提出的一个假象问题 [1]。拜占庭是东罗马帝国的首都，由于当时拜占庭罗马帝国国土辽阔，每支军队的驻地分隔很远，将军们只能靠信使传递消息。发生战争时，将军们必须制订统一的行动计划。然而，这些将军中有叛徒，叛徒希望通过影响统一行动计划的制定与传播，破坏忠诚的将军们一致的行动计划。因此，将军们必须有一个预定的方法协议，使所有忠诚的将军能够达成一致，而且少数几个叛徒不能使忠诚的将军做出错误的计划。也就是说，拜占庭将军问题的实质就是要寻找一个方法，使得将军们能在一个有叛徒的非信任环境中建立对战斗计划的共识。在分布式系统中，特别是在区块链网络环境中，也和拜占庭将军的环境类似，有运行正常的服务器（类似忠诚的拜占庭将军），有故障的服务器，还有破坏者的服务器（类似叛变的拜占庭将军）。共识算法的核心是在正常的节点间形成对网络状态的共识。

求解拜占庭将军问题，隐含要满足以下两个条件：

1）每个忠诚的将军必须收到相同的命令值 *vi*（*vi* 是第 *i* 个将军的命令）。

2）如果第 *i* 个将军是忠诚的，那么他发送的命令和每个忠诚将军收到的 *vi* 相同。

于是，拜占庭将军问题的可以描述为：一个发送命令的将军要发送一个命令给其余 $n-1$ 个将军，使得：

IC1. 所有忠诚的接收命令的将军遵守相同的命令；

IC2. 如果发送命令的将军是忠诚的，那么所有忠诚的接收命令的将军遵守所接收的命令。

Lamport 对拜占庭将军问题的研究表明 [1]，当 $n > 3m$ 时，即叛徒的个数 m 小于将军总数 n 的 1/3 时，通过口头同步通信（假设通信是可靠的），可以构造同时满足 IC1 和 IC2 的解决方案，即将军们可以达成一致的命令。但如果通信是可认证、防篡改伪造的（如采用 PKI 认证，消息签名等），则在任意多的叛徒（至少得有两个忠诚将军）的情况下都可以找到解决方案。

而在异步通信情况下，情况就没有这么乐观。Fischer-Lynch-Paterson 定理证明了，只要有一个叛徒存在，拜占庭将军问题就无解 [2]。翻译成分布式计算语言，在一个多进程异步系统中，只要有一个进程不可靠，那么就不存在一个协议，此协议能保证有限时

间内使所有进程达成一致。

由此可见，拜占庭将军问题在一个分布式系统中，是一个非常有挑战性的问题。因为分布式系统不能依靠同步通信，否则性能和效率将非常低。因此寻找一种实用的解决拜占庭将军问题的算法一直是分布式计算领域中的一个重要问题。

在这里，我们先给出分布式计算中有关拜占庭缺陷和故障的两个定义：

定义 1：拜占庭缺陷（Byzantine Fault）：

任何观察者从不同角度看，表现出不同症状的缺陷。

定义 2：拜占庭故障（Byzantine Failure）：

在需要共识的系统中由于拜占庭缺陷导致丧失系统服务。

在分布式系统中，不是所有的缺陷或故障都能称作拜占庭缺陷或故障。像死机、丢消息等缺陷或故障不能算为拜占庭缺陷或故障。拜占庭缺陷或故障是最严重缺陷或故障，拜占庭缺陷有不可预测、任意性的缺陷，例如遭黑客破坏，中木马的服务器就是一个拜占庭服务器。

在一个有拜占庭缺陷存在的分布式系统中，所有的进程都有一个初始值。在这种情况下，共识问题（Consensus Problem），就是要寻找一个算法和协议，使得该协议满足以下三个属性。

1）一致性（Agreement）：所有的非缺陷进程都必须同意同一个值。

2）正确性（Validity）：如果所有的非缺陷的进程有相同的初始值，那么所有非缺陷的进程所同意的值必须是同一个初始值。

3）可结束性（Termination）：每个非缺陷的进程必须最终确定一个值。

根据 Fischer-Lynch-Paterson 的理论，在异步通信的分布式系统中，只要有一个拜占庭缺陷的进程，就不可能找到一个共识算法，可同时满足上述要求的一致性、正确性和可结束性要求。在实际情况下，根据不同的假设条件，有很多不同的共识算法被设计出来。这些算法各有优势和局限。算法的假设条件有以下几种情况：

1）故障模型：非拜占庭故障 / 拜占庭故障。

2）通信类型：同步 / 异步。

3）通信网络连接：节点间直连数。

4）信息发送者身份：实名 / 匿名。

5）通信通道稳定性：通道可靠 / 不可靠。

6）消息认证性：认证消息 / 非认证消息。

在区块链网络中，由于应用场景的不同，所设计的目标各异，不同的区块链系统采

用了不同的共识算法。一般来说，在私有链和联盟链情况下，对一致性、正确性有很强的要求。一般来说要采用强一致性的共识算法。而在公有链情况下，对一致性和正确性通常没法做到百分之百，通常采用最终一致性（Eventual Consistency）的共识算法。

下面我们先来介绍适合私有链和联盟链场景的拜占庭容错系统。

5.1.2 拜占庭容错系统

上一节的分析表明，区块链网络的记账共识和拜占庭将军问题是相似的。参与共识记账的每一个记账节点相当于将军，节点之间的消息传递相当于信使，某些节点可能由于各种原因而产生错误的信息并传达给其他节点。通常，这些发生故障节点被称为**拜占庭节点**，而正常的节点即为**非拜占庭节点**。

拜占庭容错系统是一个拥有 n 台节点的系统，整个系统对于每一个请求，满足以下条件：

1）所有非拜占庭节点使用相同的输入信息，产生同样的结果；

2）如果输入的信息正确，那么所有非拜占庭节点必须接收这个信息，并计算相应的结果。

与此同时，在拜占庭系统的实际运行过程中，还需要假设整个系统中拜占庭节点不超过 m 台，并且每个请求还需要满足两个指标。

- 安全性：任何已经完成的请求都不会被更改，它可以在以后请求看到；
- 活性：可以接受并且执行非拜占庭客户端的请求，不会被任何因素影响而导致非拜占庭客户端的请求不能执行。

拜占庭系统普遍采用的假设条件包括：

1）拜占庭节点的行为可以是任意的，拜占庭节点之间可以共谋；

2）节点之间的错误是不相关的；

3）节点之间通过异步网络连接，网络中的消息可能丢失、乱序并延时到达，但大部分协议假设消息在有限的时间里能传达到目的地；

4）服务器之间传递的信息，第三方可以嗅探到，但是不能篡改、伪造信息的内容和验证信息的完整性。

5.1.3 实用的拜占庭容错系统

原始的拜占庭容错系统[1]由于需要展示其理论上的可行性而缺乏实用性。另外，还需要额外的时钟同步机制支持，算法的复杂度也是随节点增加而指数级增加。实用拜占

庭容错系统（Practical Byzantine Fault Tolerance，PBFT）[3]，降低了拜占庭协议的运行复杂度，从指数级别降低到多项式级别（Polynomial），使拜占庭协议在分布式系统中应用成为可能。

PBFT 是一类状态机拜占庭系统 [4]，要求共同维护一个状态，所有节点采取的行动一致。为此，需要运行三类基本协议，包括一致性协议、检查点协议和视图更换协议。我们主要关注支持系统日常运行的一致性协议。

一致性协议要求来自客户端的请求在每个服务节点上都按照一个确定的顺序执行。这个协议把服务器节点分为两类：主节点和从节点，其中主节点仅一个。在协议中，主节点负责将客户端的请求排序；从节点按照主节点提供的顺序执行请求。每个服务器节点在同样的配置信息下工作，该配置信息被称为视图，主节点更换，视图也随之变化。

一致性协议至少包含若干个阶段：请求（request）、序号分配（pre-prepare）和响应（reply）。根据协议设计的不同，可能包含相互交互（prepare），序号确认（commit）等阶段。

PBFT 的一致性协议如图 5-1 所示。PBFT 系统通常假设故障节点数为 m 个，而整个服务节点数为 $3m + 1$ 个。每一个客户端的请求需要经过 5 个阶段，通过采用两次两两交互的方式在服务器达成一致之后再执行客户端的请求。由于客户端不能从服务器端获得任何服务器运行状态的信息，PBFT 中主节点是否发生错误只能由服务器监测。如果服务器在一段时间内都不能完成客户端的请求，则会触发视图更换协议。

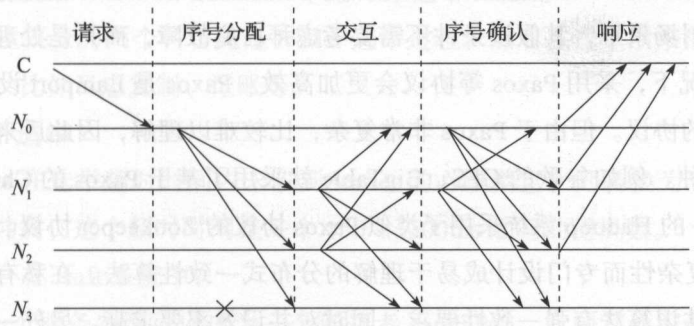


图 5-1　PBFT 协议通信模式

图 5-1 显示了一个简化的 PBFT 的协议通信模式，其中 C 为客户端，N_0 ~ N_3 表示服务节点，特别的，N_0 为主节点，N_3 为故障节点。整个协议的基本过程如下。

1）客户端发送请求，激活主节点的服务操作。

2）当主节点接收请求后，启动三阶段的协议以向各从节点广播请求。

[2.1] 序号分配阶段，主节点给请求赋值一个序列号 n，广播序号分配消息和客户端的请求消息 m，并将构造 PRE-PREPARE 消息给各从节点；

[2.2] 交互阶段，从节点接收 PRE-PREPARE 消息，向其他服务节点广播 PREPARE 消息；

[2.3] 序号确认阶段，各节点对视图内的请求和次序进行验证后，广播 COMMIT 消息，执行收到的客户端的请求并给客户端以响应。

3）客户端等待来自不同节点的响应，若有 $m+1$ 个响应相同，则该响应即为运算的结果。

PBFT 在很多场景都有应用，在区块链场景中，一般适合于对强一致性有要求的私有链和联盟链场景。例如，在 IBM 主导的区块链超级账本项目中，PBFT 是一个可选的共识协议。

除了 PBFT 之外，超级账本项目还引入了基于 PBFT 的自用共识协议，它的目的是希望在 PBFT 基础之上能够对节点的输出也做好共识，这是因为，超级账本项目的一个重要功能是提供区块链之上的智能合约，即在区块链上执行的一段代码，因此它会导致区块链账本上最终状态的不确定，为此这个自有共识协议会在 PBFT 实现的基础之上，引入代码执行结果签名进行验证。

5.1.4 Raft 协议

在很多分布式系统场景下，并不需要解决拜占庭将军问题，也就是说，在这些分布式系统的实用场景下，其假设条件不需要考虑拜占庭故障，而只是处理一般的死机故障。在这种情况下，采用 Paxos 等协议会更加高效。Paxos 是 Lamport 设计的保持分布式系统一致性的协议。但由于 Paxos 非常复杂，比较难以理解，因此后来出现了各种不同的实现和变种。例如谷歌的 GFS、BigTable 就采用了基于 Paxos 的 Chubby 的分布式锁协议；Yahoo 的 Hadoop 系统采用了类似 Paxos 协议的 Zookeeper 协议。Raft 也是为了避免 Paxos 的复杂性而专门设计成易于理解的分布式一致性算法。在私有链和联盟链的场景下，通常共识算法有强一致性要求，同时对共识效率要求高。另外一般安全性要比公有链场景高，一般来说不会经常存在拜占庭故障。因此，在一些场景下，可以考虑采用非拜占庭协议的分布式共识算法。

在 Hyperledger 的 Fabric 项目中，共识模块被设计成可插拔的模块，支持像 PBFT、Raft 等共识算法。

Raft 最初是一个用于管理复制日志的共识算法 [5]，它是一个为真实世界应用建立的

协议，主要注重协议的落地性和可理解性。Raft 是在非拜占庭故障下达成共识的强一致协议。

在区块链系统中，使用 Raft 实现记账共识的过程可以描述如下：首先选举一个 leader，接着赋予 leader 完全的权力管理记账。leader 从客户端接收记账请求，完成记账操作，生成区块，并复制到其他记账节点。有了 leader 简化了记账操作的管理。例如，leader 能够决定是否接受新的交易记录项而无需考虑其他的记账节点，leader 可能失效或与其他节点失去联系，这时，系统就会选出新的 leader。

给定 leader 方法，Raft 将共识问题分解为三个相对独立的子问题。

- leader 选举：现有的 leader 失效时，必须选出新 leader。
- 记账：leader 必须接受来自客户端的交易记录项，在参与共识记账的节点中进行复制，并使其他的记账节点认可交易所对应的区块。
- 安全：若某个记账节点对其状态机应用了某个特定的区块项，其他的服务器不能对同一个区块索引应用不同的命令。

1. Raft 基础

一个 Raft 集群通常包含 5 个服务器，允许系统有两个故障服务器。每个服务器处于 3 个状态之一：leader、follower 或 candidate。正常操作状态下，仅有一个 leader，其他的服务器均为 follower。follower 是被动的，不会对自身发出请求而是对来自 leader 和 candidate 的请求做出响应。leader 处理所有的客户端请求（若客户端联系 follower，则该 follower 将转发给 leader）。candidate 状态用来选举 leader。

Raft 阶段主要分为两个，首先是 leader 选举过程，然后在选举出来的 leader 基础上进行正常操作，比如日志复制、记账等。

2. leader 选举

当 follower 在选举超时时间内未收到 leader 的心跳消息，则转换为 candidate 状态。为了避免选举冲突，这个超时时间是一个 150 ~ 300ms 之间的随机数。

一般而言，在 Raft 系统中：

1）任何一个服务器都可以成为一个候选者 candidate，它向其他服务器 follower 发出要求选举自己的请求。

2）其他服务器同意了，发出 OK。注意，如果在这个过程中，有一个 follower 宕机，没有收到请求选举的要求，此时候选者可以自己选自己，只要达到 N/2 + 1 的大多数票，候选人还是可以成为 leader 的。

3）这样这个候选者就成为了 leader 领导人，它可以向选民也就是 follower 发出指

令，比如进行记账。

4）以后通过心跳进行记账的通知。

5）一旦这个 leader 崩溃了，那么 follower 中有一个成为候选者，并发出邀票选举。

6）follower 同意后，其成为 leader，继续承担记账等指导工作。

3. 记账过程

Raft 的记账过程按以下步骤完成：

1）假设 leader 领导人已经选出，这时客户端发出增加一个日志的要求；

2）leader 要求 follower 遵从他的指令，都将这个新的日志内容追加到他们各自日志中；

3）大多数 follower 服务器将交易记录写入账本后，确认追加成功，发出确认成功信息；

4）在下一个心跳中，leader 会通知所有 follower 更新确认的项目。

对于每个新的交易记录，重复上述过程。

如果在这一过程中，发生了网络通信故障，使得 leader 不能访问大多数 follower 了，那么 leader 只能正常更新它能访问的那些 follower 服务器。而大多数的服务器 follower 因为没有了 leader，他们将重新选举一个候选者作为 leader，然后这个 leader 作为代表与外界打交道，如果外界要求其添加新的交易记录，这个新的 leader 就按上述步骤通知大多数 follower，如果这时网络故障修复了，那么原先的 leader 就变成 follower，在失联阶段，这个老 leader 的任何更新都不能算确认，都回滚，接收新的 leader 的新的更新。

本节介绍了分布式系统中的常用共识算法。从介绍拜占庭将军问题开始，介绍了拜占庭容错系统、状态机拜占庭协议、实用拜占庭容错协议（PBFT）和 Raft。其中拜占庭容错协议和 Raft 是联盟链和私有链上常用的共识算法。

而公共链的共识机制一般采用工作量证明（POW）和权益证明（POS）算法。下面进行介绍。

5.2 PoW 机制

比特币系统的重要概念是一个基于互联网的去中心化账本，即区块链，每个区块相当于账本页，区块中记录的信息主体，即为相应的交易内容。账本内容的唯一性要求记账行为是中心化的行为，然而，中心化所引发的单点失败，可能导致整个系统面临危机

甚至崩溃。去中心记账可以克服中心化账本的弱点，但同时也会带来记账行为的一致性问题。

从去中心化账本系统的角度看，每个加入这个系统的节点都要保存一份完整的账本，但每个节点却不能同时记账，因为节点处于不同的环境，接收到不同的信息，如果同时记账的话，必然会导致账本的不一致，造成混乱。因此，需要有共识来达成哪个节点有权记账。比特币区块链通过竞争记账的方式解决去中心化的记账系统的一致性问题。

比特币系统设计了以每个节点的计算能力即“算力”来竞争记账权的机制。在比特币系统中，大约每 10 分钟进行一轮算力竞赛，竞赛的胜利者，就获得一次记账的权力，并向其他节点同步新增账本信息。

然而，在一个去中心化的系统中，谁有权判定竞争的结果呢？比特币系统是通过一个称为“工作量证明”(Proof of Work，PoW）的机制完成的。

简单地说，PoW 就是一份确认工作端做过一定量工作的证明。PoW 系统的主要特征是计算的不对称性。工作端需要做一定难度的工作得出一个结果，验证方却很容易通过结果来检查工作端是不是做了相应的工作。

举个例子，给定字符串“ blockchain”，我们给出的工作量要求是，可以在这个字符串后面连接一个称为 nonce 的整数值串，对连接后的字符串进行 SHA256 哈希运算，如果得到的哈希结果（以十六进制的形式表示）是以若干个 0 开头的，则验证通过。为了达到这个工作量证明的目标，我们需要不停地递增 nonce 值，对得到的新字符串进行 SHA256 哈希运算。按照这个规则，需要经过 2688 次计算才能找到前 3 位均为 0 的哈希值，而要找到前 6 位均为 0 的哈希值，则需进行 620 969 次计算。

```
blockchain1 → 4bfb943cba9fb9926df93f33c17d64b378d56714e8a29c6ba8bdc9690cea8e27
blockchain2 → 01181212a283e760929f6b1628d903127c65e6fb5a9ad7fe94b790e699269221
......
blockchain515 → 0074448bea8027bebd6333d3aa12fd11641e051911c5bab661a9b849b83958a7
......
blockchain2688 → 0009b257eb8cf9eba179ab2be74d446fa1c59f0adfa8814260f52ae0016dd50f
......
blockchain48851: 00000b3d96b4db1a976d3a69829aabef8bafa35ab5871e084211a16d3a4f385c
......
blockchain6200969: 000000db7fa334aef754b51792cff6c880cd286c5f490d5cf73f658d9576d424
```

通过上面这个计算特定 SHA256 运算结果的示例，我们对 PoW 机制有了一个初步的理解。对于特定字符串后接随机 nonce 值所构成的串，要找到这样的 nonce 值，满足

前 n 位均为 0 的 SHA256 值，需要多次进行哈希值的计算。一般来说，n 值越大，需要完成的哈希计算量也越大。由于哈希值的伪随机特性，要寻找 4 个前导 0 的哈希值，预期大概要进行 2^{16} 次尝试，这个数学期望的计算次数，就是所要求的“工作量”。

比特币网络中任何一个节点，如果想生成一个新的区块并写入区块链，必须解出比特币网络出的 PoW 问题。这道题关键的 3 个要素是工作量证明函数、区块及难度值。工作量证明函数是这道题的计算方法，区块决定了这道题的输入数据，难度值决定了这道题所需要的计算量。

1. 工作量证明函数

比特币系统中使用的工作量证明函数是 SHA256。

SHA 是安全哈希算法（Secure Hash Algorithm）的缩写，是一个密码哈希函数家族。这一组函数是由美国国家安全局（NSA）设计，美国国家标准与技术研究院（NIST）发布的，主要适用于数字签名标准。SHA256 就是这个函数家族中的一个，是输出值为 256 位的哈希算法。到目前为止，还没有出现对 SHA256 算法的有效攻击。具体见第 4 章的讲解。

2. 区块

比特币的区块由区块头及该区块所包含的交易列表组成。区块头的大小为 80 字节，由 4 字节的版本号、32 字节的上一个区块的哈希值、32 字节的 Merkle 根哈希值、4 字节的时间戳（当前时间）、4 字节的当前难度值、4 字节的随机数组成。区块包含的交易列表则附加在区块头后面，其中的第一笔交易是 coinbase 交易，这是一笔为了让矿工获得奖励及手续费的特殊交易。

区块的大致结构如图 5-2 所示。

版本	020 000 00
前一区块链（反向）	17975b97c18ed1f7e255adf297599b55330edab87803c81701000000000000000
树（反向）	8a97295s274764f1a0b3948df3990344c0e19fa6b2b92b3a19c8e6vadc141787
时间戳	358b0553
比特	533f0119
随机数	48750833
交易计数	63
铸币库交易	
交易	
…	

图 5-2 区块的结构

拥有 80 字节固定长度的区块头，就是用于比特币工作量证明的输入字符串。因此，为了使区块头能体现区块所包含的所有交易，在区块的构造过程中，需要将该区块要包含的交易列表，通过 Merkle 树算法生成 Merkle 根哈希值，并以此作为交易列表的哈希值存到区块头中。其中 Merkle 树的算法图解如图 5-3 所示。

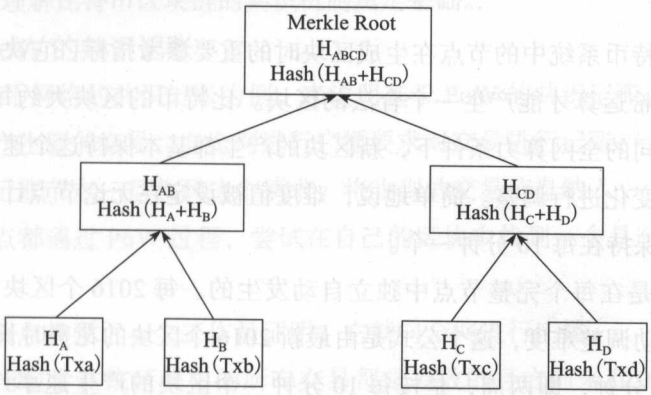


图 5-3　带 4 个交易记录的 Merkle 树根哈希值的计算

图 5-3 展示了一个具有 4 个交易记录的 Merkle 树的根哈希值的计算过程。首先以这 4 个交易作为叶子结点构造一棵完全二叉树，然后通过哈希值的计算，将这棵二叉树转化为 Merkle 树。

首先对 4 个交易记录：Txa ~ Txc，分别计算各自的哈希值 H_A ~ H_C，然后计算两个中间节点的哈希值 H_{AB} = Hash(H_A+H_B) 和 H_{CD} = Hash(H_C+H_D)，最后计算出根节点的哈希值 H_{ABCD} = Hash(H_{AB}+H_{CD})。

而构造出来的区块链呈现如图 5-4 所示。

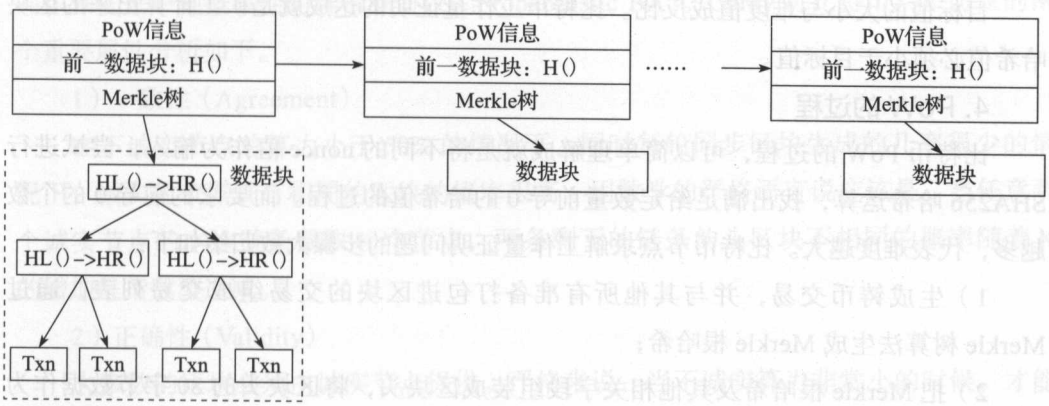


图 5-4　区块链的简化结构

由图 5-4 所示的简化的区块链结构，我们可以发现，所有在给定时间范围需要记录的交易信息被构造成一个 Merkle 树，区块中包含了指向这个 Merkle 树的哈希指针，关联了与该区块相关的交易数据，同时，区块中也包含了指向前一区块的哈希指针，使得记录了不同交易的单个区块被关联起来，形成区块链。

3. 难度值

难度值是比特币系统中的节点在生成区块时的重要参考指标，它决定了节点大约需要经过多少次哈希运算才能产生一个合法的区块。比特币的区块大约每 10 分钟生成一个，如果要在不同的全网算力条件下，新区块的产生都基本保持这个速率，难度值必须根据全网算力的变化进行调整。简单地说，难度值被设定在无论节点计算能力如何，新区块产生速率都保持在每 10 分钟一个。

难度的调整是在每个完整节点中独立自动发生的。每 2016 个区块，所有节点都会按统一的公式自动调整难度，这个公式是由最新 2016 个区块的花费时长与期望时长（期望时长为 20 160 分钟，即两周，是按每 10 分钟一个区块的产生速率计算出的总时长）比较得出的，根据实际时长与期望时长的比值，进行相应调整（或变难或变易）。也就是说，如果区块产生的速率比 10 分钟快则增加难度，比 10 分钟慢则降低难度。

这个公式可以总结为如下形式：

新难度值 = 旧难度值 ×(过去 2016 个区块花费时长 /20 160 分钟)

工作量证明需要有一个目标值。比特币工作量证明的目标值（Target）的计算公式如下：

目标值 = 最大目标值 / 难度值

其中最大目标值为一个恒定值：

0x00000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

目标值的大小与难度值成反比。比特币工作量证明的达成就是矿工计算出来的区块哈希值必须小于目标值。

4. PoW 的过程

比特币 PoW 的过程，可以简单理解成就是将不同的 nonce 值作为输入，尝试进行 SHA256 哈希运算，找出满足给定数量前导 0 的哈希值的过程。而要求的前导 0 的个数越多，代表难度越大。比特币节点求解工作量证明问题的步骤大致归纳如下：

1）生成铸币交易，并与其他所有准备打包进区块的交易组成交易列表，通过 Merkle 树算法生成 Merkle 根哈希；

2）把 Merkle 根哈希及其他相关字段组装成区块头，将区块头的 80 字节数据作为工作量证明的输入；

3）不停地变更区块头中的随机数，即 nonce 的数值，并对每次变更后的区块头做双重 SHA256 运算（即 SHA256(SHA256(Block_Header))），将结果值与当前网络的目标值做对比，如果小于目标值，则解题成功，工作量证明完成。

比特币的工作量证明，就是俗称“挖矿”所做的主要工作。理解工作量证明机制，将为我们进一步理解比特币区块链的共识机制奠定基础。

5. 基于 PoW 的共识记账

我们以比特币网络的共识记账为例，来说明基于 PoW 的共识记账过程。

1）客户端产生新的交易，向全网进行广播要求对交易进行记账；

2）每一个记账节点一旦收到这个请求，将收到的交易信息纳入一个区块中；

3）每个节点都通过 PoW 过程，尝试在自己的区块中找到一个具有足够难度的工作量证明；

4）当某个节点找到了一个工作量证明，它就向全网进行广播；

5）当且仅当包含在该区块中的所有交易都是有效的且之前未存在过的，其他节点才认同该区块的有效性；

6）其他节点表示它们接受该区块，而表示接受的方法则是在跟随该区块的末尾，制造新的区块以延长该链条，而将被接受区块的随机哈希值视为先于新区块的随机哈希值。

通过上述的记账过程，客户端所要求记录的交易信息被写入了各个记账节点的区块链中，形成了一个分布式的高概率的一致账本。

6. 关于比特币 PoW 能否解决拜占庭将军的问题

关于比特币 PoW 共识机制能否解决拜占庭将军问题一直在业界有争议。2015 年，Juan Garay 对比特币的 PoW 共识算法进行了正式的分析，得出的结论是比特币的 PoW 共识算法是一种概率性的拜占庭协议（Probabilistic BA）。Garay 对比特币共识协议的两个重要属性分析如下。

1）一致性（Agreement）

在不诚实节点总算力小于 50% 的情况下，同时每轮同步区块生成的几率很少的情况下，诚实的节点具有相同的区块的概率很高。用数学的严格语言说应该是：当任意两个诚实节点的本地链条截取 K 个节点，两条剩下的链条的头区块不相同的概率随着 K 的增加呈指数型递减。

2）正确性（Validity）

大多数的区块必须由诚实节点提供。严格来说，当不诚实算力非常小的时候，才能使大多数区块由诚实节点提供。

因此可以看到，当不诚实的算力小于网络总算力的 50% 时，同时挖矿难度比较高，在大约 10 分钟出一个区块情况下，比特币网络达到一致性的概念会随确认区块的数目增多而呈指数型增加。但当不诚实算力具一定规模，甚至不用接近 50% 的时候，比特币的共识算法并不能保证正确性，也就是，不能保证大多数的区块由诚实节点来提供。

因此，我们可以看到，比特币的共识算法不适合于私有链和联盟链。其原因首先是它是一个最终一致性共识算法，不是一个强一致性共识算法。第二个原因是其共识效率低。提供共识效率又会牺牲共识协议的安全性。

另一方面，比特币通过巧妙的矿工奖励机制来提升网络的安全性。矿工挖矿获得比特币奖励以及记账所得的交易费用使得矿工更希望维护网络的正常运行，而任何破坏网络的非诚信行为都会损害矿工自身的利益。因此，即使有些比特币矿池具备强大的算力，它们都没有作恶的动机，反而有动力维护比特币的正常运行，因为这和它们的切实利益相关。

5.3 PoS 机制

PoW 背后的基本概念很简单：工作端提交已知难于计算但易于验证的计算结果，而其他任何人都能够通过验证这个答案就确信工作端为了求得结果已经完成了量相当大的计算工作。然而 PoW 机制存在明显的弊端。一方面，PoW 的前提是，节点和算力是均匀分布的，因为通过 CPU 的计算能力来进行投票，拥有钱包（节点）数和算力值应该是大致匹配的，然而随着人们将 CPU 挖矿逐渐升级到 GPU、FPGA，直至 ASIC 矿机挖矿，节点数和算力值也渐渐失配。另一方面，PoW 太浪费了。比特币网络每秒可完成数百万亿次 SHA256 计算，但这些计算除了使恶意攻击者不能轻易地伪装成几百万个节点和打垮比特币网络，并没有更多实际或科学价值。当然，相对于允许世界上任何一个人在瞬间就能通过去中心化和半匿名的全球货币网络，给其他人几乎没有手续费地转账所带来的巨大好处，它的浪费也许只算是很小的代价。

有鉴于此，人们提出了一些工作量证明的替代者。权益证明（Proof of Stake，PoS）就是其中的一种方法。

权益证明要求用户证明拥有某些数量的货币（即对货币的权益），点点币（Peercoin）是首先采用权益证明的货币，尽管它依然使用工作量证明挖矿。

1. PoS 的应用

点点币在 SHA256 的哈希运算的难度方面引入了币龄的概念，使得难度与交易输入

的币龄成反比。在点点币中，币龄被定义为币的数量与币所拥有的天数的乘积，这使得币龄能够反映交易时刻用户所拥有的货币数量。

实际上，点点币的权益证明机制结合了随机化与币龄的概念，未使用至少 30 天的币可以参与竞争下一区块，越久和越大的币集有更大的可能去签名下一区块。然而，一旦币的权益被用于签名一个区块，则币龄将清为零，这样必须等待至少 30 日才能签署另一区块。同时，为防止非常老或非常大的权益控制区块链，寻找下一区块的最大概率在 90 天后达到最大值，这一过程保护了网络，并随着时间逐渐生成新的币而无需消耗大量的计算能力。点点币的开发者声称这将使得恶意攻击变得困难，因为没有中心化的挖矿池需求，而且购买半数以上的币的开销似乎超过获得 51% 的工作量证明的哈希计算能力。

权益证明必须采用某种方法定义任意区块链中的下一合法区块，依据账户结余来选择将导致中心化，例如单个首富成员可能会拥有长久的优势。为此，人们还设计了其他不同的方法来选择下一合法区块。

2. 随机区块选择

NXT 币和黑币采用随机方法预测下一合法区块，使用公式查找与权益大小结合的最小哈希值。由于权益公开，每个节点都可以合理的准确度预计哪个账户有权建立区块。

3. 基于权益速度的选择

瑞迪币（Reddcoin）引入权益速度证明，即鼓励钱币的流动而非囤积。通过给币龄引入指数衰减函数，使得 1 币的币龄不会超过 2 币月。

5.4　DPoS 机制

PoW 机制和 PoS 机制虽然都能有效地解决记账行为的一致性共识问题，但是现有的比特币 PoW 机制纯粹依赖算力，导致专业从事挖矿的矿工群体似乎已和比特币社区完全分隔，某些矿池的巨大算力俨然成为另一个中心，这与比特币的去中心化思想相冲突。PoS 机制虽然考虑到了 PoW 的不足，但依据权益结余来选择，会导致首富账户的权力更大，有可能支配记账权。股份授权证明机制（Delegated Proof of Stake，DPoS）的出现正是基于解决 PoW 机制和 PoS 机制的这类不足。

比特股（Bitshare）是一类采用 DPoS 机制的密码货币，它期望通过引入一个技术民主层来减少中心化的负面影响。

比特股引入了见证人这个概念，见证人可以生成区块，每一个持有比特股的人都可以投票选举见证人。得到总同意票数中的前 N 个（N 通常定义为 101）候选者可以当选

为见证人，当选见证人的个数（N）需满足：至少一半的参与投票者相信 N 已经充分地去中心化。

见证人的候选名单每个维护周期（1 天）更新一次。见证人然后随机排列，每个见证人按序有 2 秒的权限时间生成区块，若见证人在给定的时间片不能生成区块，区块生成权限交给下一个时间片对应的见证人。DPoS 的这种设计使得区块的生成更为快速，也更加节能。

DPoS 充分利用了持股人的投票，以公平民主的方式达成共识，他们投票选出的 N 个见证人，可以视为 N 个矿池，而这 N 个矿池彼此的权利是完全相等的。持股人可以随时通过投票更换这些见证人（矿池），只要他们提供的算力不稳定，计算机宕机，或者试图利用手中的权力作恶。

比特股还设计了另外一类竞选，代表竞选。选出的代表拥有提出改变网络参数的特权，包括交易费用、区块大小、见证人费用和区块区间。若大多数代表同意所提出的改变，持股人有两周的审查期，这期间可以罢免代表并废止所提出的改变。这一设计确保代表技术上没有直接修改参数的权利以及所有的网络参数的改变最终需得到持股人的同意。

5.5 Ripple 共识算法

1. Ripple 的网络结构

Ripple（瑞波）是一种基于互联网的开源支付协议，可以实现去中心化的货币兑换、支付与清算功能。在 Ripple 的网络中，交易由客户端（应用）发起，经过追踪节点（tracking node）或验证节点（validating node）把交易广播到整个网络中。追踪节点的主要功能是分发交易信息以及响应客户端的账本请求。验证节点除包含追踪节点的所有功能外，还能够通过共识协议，在账本中增加新的账本实例数据。如图 5-5 所示是 Ripple 的共识过程中节点交互示意图。

2. Ripple 共识算法

Ripple 的共识达成发生在验证节点之间，每个验证节点都预先配置了一份可信任节点名单，称为 UNL（Unique Node List）。在名单上的节点可对交易达成进行投票。每隔几秒，Ripple 网络将进行如下共识过程：

1）每个验证节点会不断收到从网络发送过来的交易，通过与本地账本数据验证后，不合法的交易直接丢弃，合法的交易将汇总成交易候选集（candidate set）。交易候选集里面还包括之前共识过程无法确认而遗留下来的交易。

2）每个验证节点把自己的交易候选集作为提案发送给其他验证节点。

3）验证节点在收到其他节点发来的提案后，如果不是来自 UNL 上的节点，则忽略该提案；如果是来自 UNL 上的节点，就会对比提案中的交易和本地的交易候选集，如果有相同的交易，该交易就获得一票。在一定时间内，当交易获得超过 50% 的票数时，则该交易进入下一轮。没有超过 50% 的交易，将留待下一次共识过程去确认。

4）验证节点把超过 50% 票数的交易作为提案发给其他节点，同时提高所需票数的阈值到 60%，重复步骤 3）、步骤 4），直到阈值达到 80%。

5）验证节点把经过 80%UNL 节点确认的交易正式写入本地的账本数据中，称为最后关闭账本（Last Closed Ledger），即账本最后（最新）的状态。

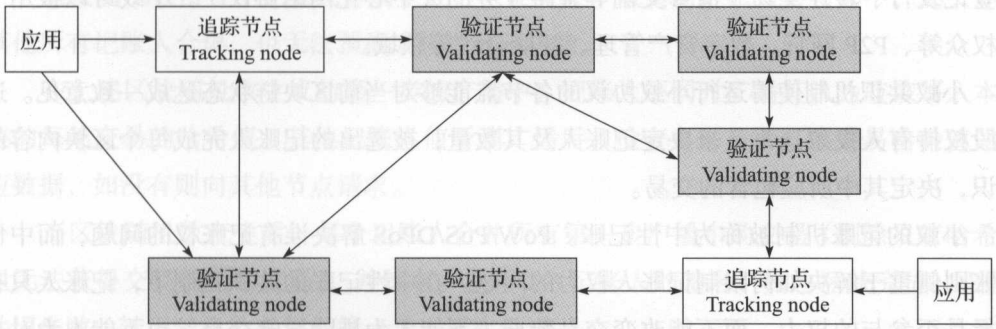


图 5-5　Ripple 共识过程节点交互示意图

图 5-6 是 Ripple 共识算法流程。

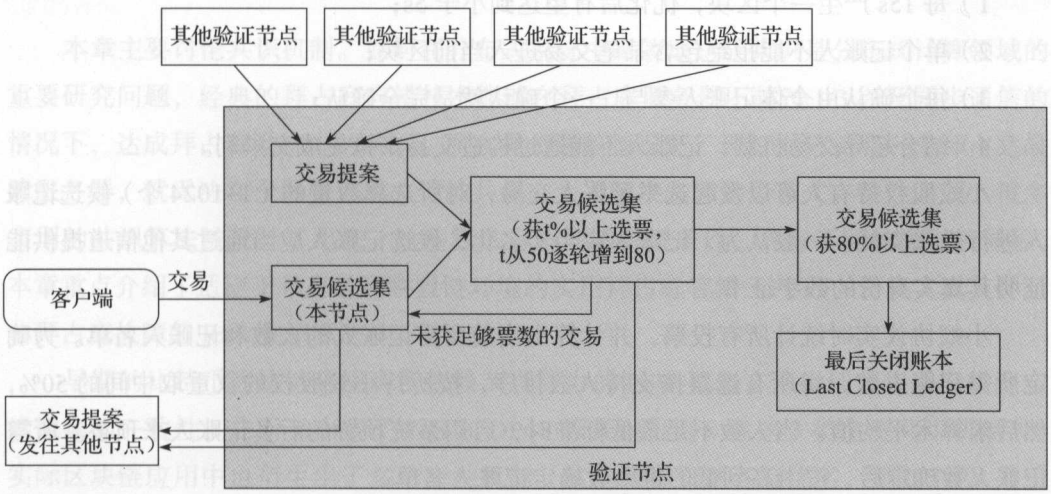


图 5-6　Ripple 共识算法流程

在 Ripple 的共识算法中，参与投票节点的身份是事先知道的，因此，算法的效率比 PoW 等匿名共识算法要高效，交易的确认时间只需几秒钟。当然，这点也决定了该共识算法只适合于权限链（Permissioned chain）的场景。Ripple 共识算法的拜占庭容错（BFT）能力为（n–1）/5，即可以容忍整个网络中 20% 的节点出现拜占庭错误而不影响正确的共识。

5.6 小蚁共识机制

小蚁是基于区块链技术，将实体世界的资产和权益进行数字化，通过点对点网络进行登记发行、转让交易、清算交割等金融业务的去中心化网络协议 [7]。小蚁可以被用于股权众筹、P2P 网贷、数字资产管理、智能合约等领域。

小蚁共识机制使得运行小蚁协议的各节点能够对当前区块链状态达成一致意见。通过股权持有人投票选举，来决定记账人及其数量；被选出的记账人完成每个区块内容的共识，决定其中所应包含的交易。

小蚁的记账机制被称为中性记账。PoW/PoS/DPoS 解决谁有记账权的问题，而中性记账则侧重于解决如何限制记账人权力的问题。在中性记账的共识机制下，记账人只有选择是否参与的权力，而不能改变交易数据，不能人为排除某笔交易，也不能人为对交易进行排序。

小蚁的中性记账区块链可以做到：

1）每 15s 产生一个区块，优化后有望达到小于 5s；

2）单个记账人不能拒绝包含某笔交易进入当前区块；

3）每个确认由全体记账人参与，一个确认就是完全确认；

4）结合超导交易机制，记账人不能通过构造交易来抢先成交牟利。

小蚁股权持有人可以发起选举记账人交易，对所选择数量的（1~1024 个）候选记账人进行投票支持。一般认为，记账人应当实名化，候选记账人应当通过其他信道提供能证明其真实身份的数字证书。

小蚁协议实时统计所有投票，并计算出当前所需记账人的人数和记账人名单。为确定所需记账人数，将所有选票按支持人数排序，按所持小蚁股权的权重取中间的 50%，然后求算术平均值。当人数不足最低标准时，启用系统预置的后备记账人来顶替。所需记账人数确定后，按由高到低的得票数确定记账人名单。

我们以区块随机数的生成来了解小蚁共识机制。每个区块生成前，记账人之间需

要协作生成一个区块随机数。小蚁使用沙米尔秘密共享方案（Shamir's Secret Sharing Scheme，SSSS）来协作生成随机数。

依据 SSSS 方案，可以将密文 S 生成 N 份密文碎片，持有其中的 K 份，就能还原出密文 S。小蚁记账人（假设为 N+1 个）之间通过以下 3 步对随机数达成共识：

1）自选一个随机数，将此随机数通过 SSSS 方案生成 N 份碎片，用其他 N 个记账人的公钥加密，并广播。

2）收到其他 N 个记账人的广播后，将其中自己可解密的部分解密，并广播。

3）收集到至少 K 份密文碎片后，解出随机数；获得所有记账人的随机数后，合并生成区块随机数。

区块随机数由各个记账人协同生成，只要有一个诚实的记账人参与其中，那么即便其他所有记账人合谋，也无法预测或构造此随机数。

在上述区块随机数生成的第一步的广播中，记账人还同时广播其认为应该写入本区块的每笔交易的哈希值。其他记账人侦听到广播后，检查自己是否有该交易哈希值的对应数据，如没有则向其他节点请求。

当区块随机数产生后，每个记账人合并所有第一步广播中的交易（剔除只有哈希值但无法获得交易数据的交易），并签名。获得 2/3 记账人的签名，则本区块完成；否则，共识失败，转回随机数共识的第一步，再次尝试。

5.7　本章小结

本章主要讨论共识机制。如何在分布式系统中高效地达成共识是分布式计算领域的重要研究问题，经典的拜占庭容错技术能够在拜占庭服务器不超过 1/3 以及同步通信的情况下，达成拜占庭系统中的共识。而在异步通信情况下，理论上只要有一个拜占庭故障服务器，就无法在全网中达成一致的共识。为了解决实际的分布式一致性问题，很多实用的共识算法被设计了出来。这些算法有不同的假设条件，具有不同的优点和局限。本章重点介绍了适应于私有链和联盟链环境的实用拜占庭容错（PBFT）协议，以及针对非拜占庭故障的 Raft 共识算法。

早期的比特币区块链采用高度依赖节点算力的 PoW 机制，来保证比特币网络分布式记账的一致性，之后又出现了 PoS 和 DPoS 等共识机制。除这 3 类主流共识机制外，实际区块链应用中也衍生出了多个变种机制。这些共识机制各有优劣。例如 PoW 共识机制在安全性和公平性上比较有优势，也依靠其先发优势已经形成成熟的挖矿产业链，

但也因为其对能源的消耗而饱受诟病。而新兴的机制，如 PoS 和 DPoS 等则更为环保和高效，但在安全性和公平性方面比不上 PoW 机制。

一般来说，PoW 和 PoS 机制比较适合公共链环境，而 PBFT 和 Raft 则比较适合联盟链和私有链的分布式环境。比特币的 PoW 机制是一种概念性的拜占庭协议，能在一定程度上解决拜占庭问题。而 PoS 等其他机制，目前并没有严格的分析证明其在拜占庭协议方面的属性。

参考资料

[1] Lamport L, Shostak R, Pease M. The Byzantine generals problem. ACM Trans. on Programming Languages and Systems, 1982, 4(3): 382-401.

[2] Fischer, M.J., Lynch, N.A., Paterson, M.: Impossibility of distributed consensus with one faulty process. J. ACM 32(2), 374-382 (1985).

[3] Castro M, Liskov B. Practical Byzantine fault tolerance and proactive recovery. ACM Trans. on Computer Systems, 2002, 20(4): 398-461.

[4] 范捷，易乐天，舒继武．拜占庭系统技术研究综述．软件学报，2013，24(6)：1346-1360.

[5] Ongaro D, Ousterhout J . In Search of an Understandable Consensus Algorithm. In: Proc. of USENIX Annual Technical Conference 2014, 305-319.

[6] Garay, J.A., Kiayias, A., Leonardos, N.: The Bitcoin Backbone Protocol: Analysis and Applications. IACR Cryptology ePrint Archive 2014, 765 (2014).

[7] 小蚁白皮书 https://www.antshares.org/Files/ 小蚁白皮书 %201.0.htm.

第6章

比特币应用开发指南

本章在常见的 Ubuntu14.04 Desktop 64bit 操作系统上，采用 Docker 容器技术来快速安装和配置私有节点，用比特币测试网络（bitcoin-testnet）作为开发试验环境，以 Node.js 程序语言为例子，说明如何调用比特币钱包节点提供的 RPC 接口服务，实现一些涉及比特币区块链的具体应用功能。

RPC（Remote Procedure Call）即远程过程调用协议，是一种通过网络从远程计算机程序上请求服务，而不需要了解底层网络技术的协议。

在 Bitcoin 的 Wiki 网页上面（https://en.bitcoin.it/wiki/API_reference_(JSON-RPC)）可以看到，除了 Node.js 外还有很多种语言都可以调用 Bitcoin 的 RPC，读者可以参考本章内容选择适合自己的语言具体试验。

在上面的网页里，还可以延伸阅读和了解 Bitcoin RPC 能调用的命令列表（https://en.bitcoin.it/wiki/Original_Bitcoin_client/API_calls_list）。

6.1 以虚拟机方式搭建应用开发环境

这里我们以常用的 Microsoft Windows 7 64bit 桌面操作系统为例，安装 Oracle VM VirtualBox 虚拟机支持软件，来进一步安装 Ubuntu14.04 Desktop 64bit 版操作系统，作为基础开发环境。

6.1.1 下载和安装 Oracle VM VirtualBox

Oracle VM VirtualBox 是一款开源的虚拟机软件。对于需要跨不同操作环境开发代

码的开发人员是一个很有用的工具。由于 VirtualBox 允许在一台计算机上运行多个虚拟操作系统（如 Solaris、Windows、DOS、Linux、OS/2 Warp、BSD 等），开发人员只需在不同桌面窗口之间进行切换即可轻松切换操作系统。

1）从网址 https://www.virtualbox.org/wiki/Downloads 下载 VirtualBox。

从网页里选择 VirtualBox 5.1.2 for windows hosts x86/amd64 版本，单击下载，如图 6-1 所示。

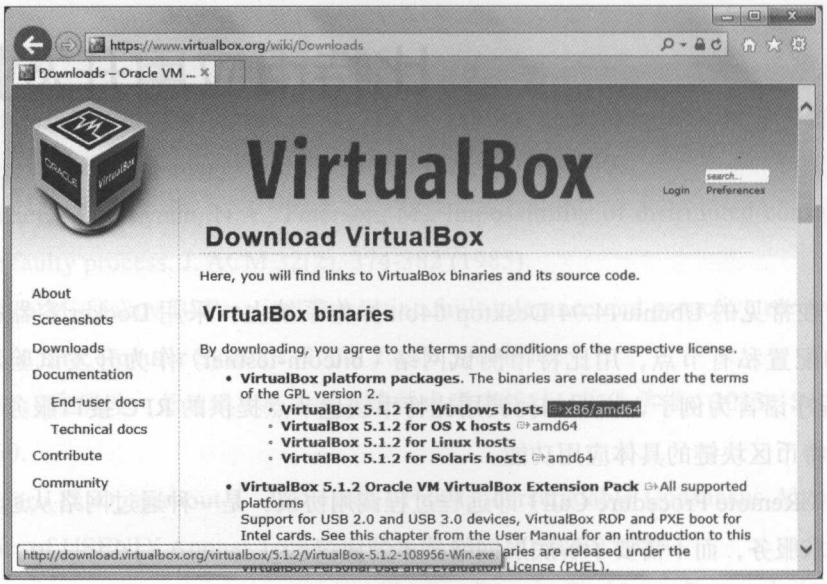


图 6-1　选择合适版本下载

2）下载完成后，双击安装文件，开始如图 6-2 所示的安装步骤。

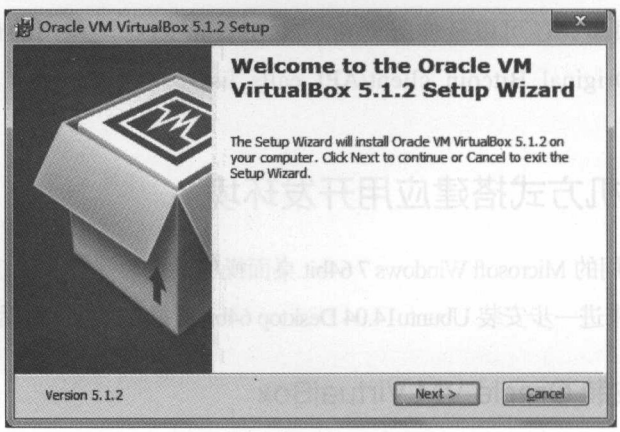


图 6-2　安装向导启动

单击 Next 按钮继续下一步操作，进行安装组件和路径选择，如图 6-3 所示。

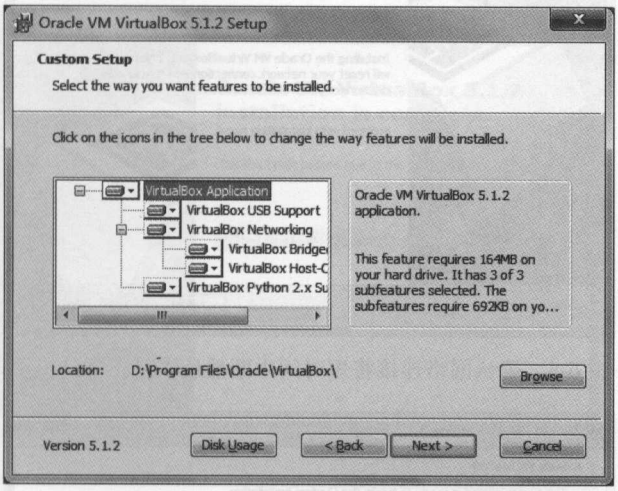


图 6-3　安装组件和路径选择

选择安装位置后（注意，安装路径中不能包含中文字符），单击 Next 按钮继续下一步操作，如图 6-4 所示。

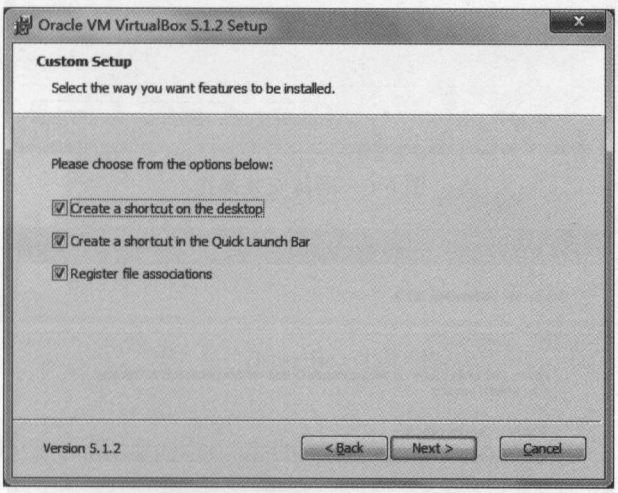


图 6-4　选择是否创建快捷方式

单击 Next 按钮后，继续如图 6-5 所示的下一步操作。

单击 Yes 按钮后继续下一步确认安装操作，如图 6-6 所示。

单击 Install 按钮继续下一步操作，如图 6-7 所示。

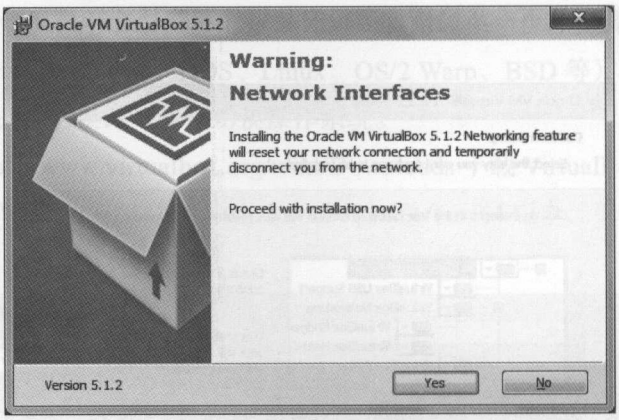


图 6-5 确认网络连接将短时间中断然后恢复正常的提示

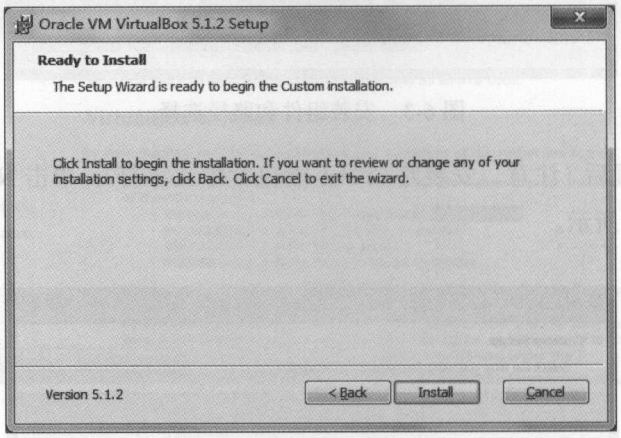


图 6-6 确认安装操作

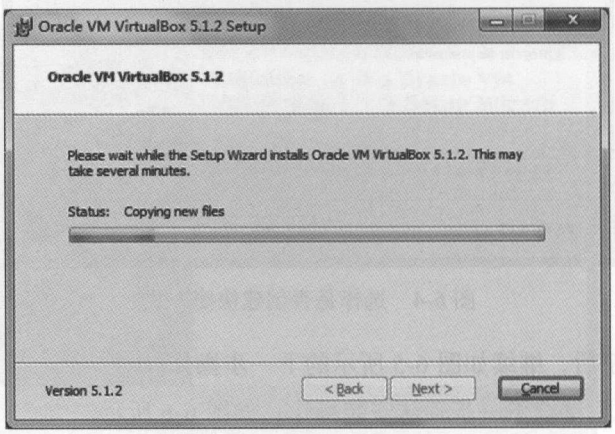


图 6-7 安装进度显示

单击 Next 按钮后进行如图 6-8 所示的下一步操作。

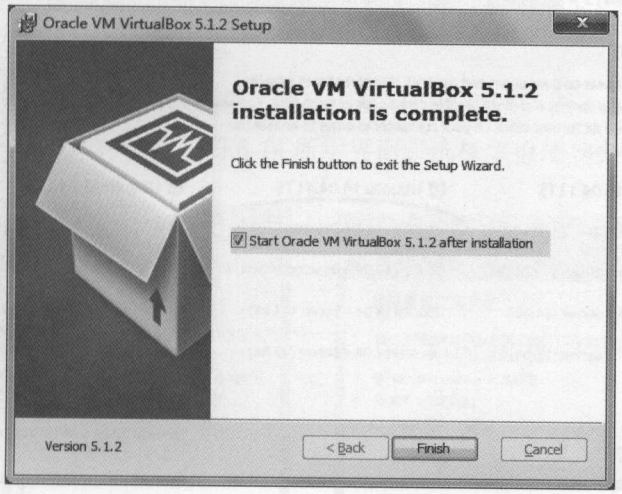


图 6-8　安装完成

单击 Finish 按钮后进入 VirtualBox 主界面，如图 6-9 所示。

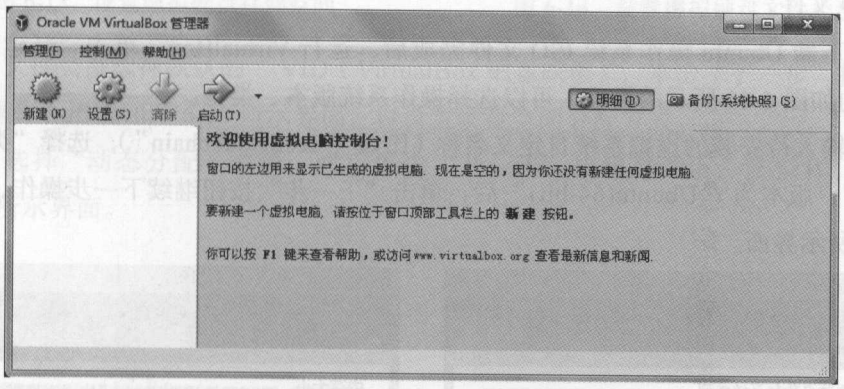


图 6-9　VirtualBox 主界面

6.1.2　以虚拟机方式安装 Ubuntu14.04

1）从下述网址下载 Ubuntu 14.04 Desktop 64bit 的 BT 种子文件。

```
http://www.ubuntu.com/download/alternative-downloads
```

从网页里选择 BitTorrent 下方的 Ubuntu 14.04.4 Desktop(64-bit) 版本，单击下载，如图 6-10 所示。

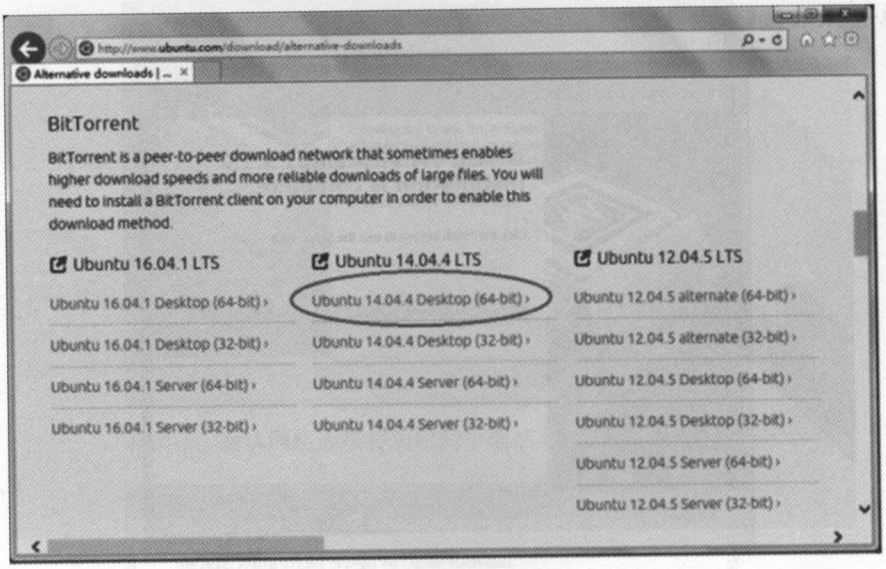


图 6-10　选择合适版本的 BT 种子文件下载

2）用 BT 下载软件打开刚下载的 BT 种子文件，下载对应的 Ubuntu 操作系统光盘镜像 ISO 文件。

3）下载 Ubuntu 操作系统 ISO 文件完成后，运行 VirtualBox，单击“新建”按钮，在出现的如图 6-11 所示的界面，可以选择操作系统版本、类型。

4）输入待安装的虚拟系统自定义名称（比如“ TestBlockchain”），选择“类型”为“ Linux”、版本为“ Ubuntu(64-bit)”后，单击“下一步”按钮继续下一步操作，出现如图 6-12 所示界面。

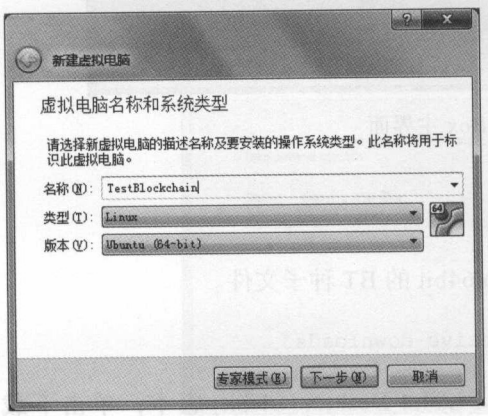


图 6-11　选择安装虚拟操作系统版本、类型

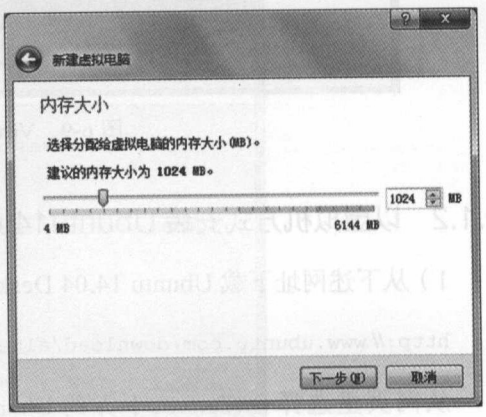


图 6-12　设置内存大小

5）在图 6-12 所示界面里选择虚拟系统运行内存的大小，一般按建议的默认内存值设定即可，根据实际机器的内存大小也可稍作调整。然后单击“下一步”按钮继续下一步操作，出现如图 6-13 所示界面。

6）对于新建虚拟系统，默认选择“现在创建虚拟硬盘”（见图 6-13），然后单击“创建”按钮继续下一步操作，出现如图 6-14 所示界面，选择虚拟硬盘文件类型。

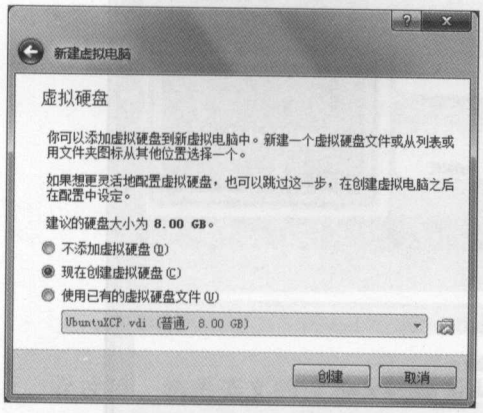


图 6-13　设置虚拟硬盘存储空间

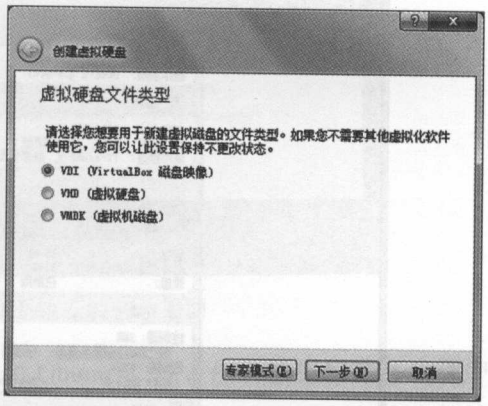


图 6-14　设置虚拟硬盘文件类型

7）文件类型选择默认的“ VID（VirtualBox 磁盘映像）”，单击“下一步”按钮继续下一步操作，出现如图 6-15 所示界面，以设置硬盘空间分配方式。

8）选择“动态分配”选项，然后单击“下一步”按钮继续下一步操作，出现如图 6-16 所示界面。

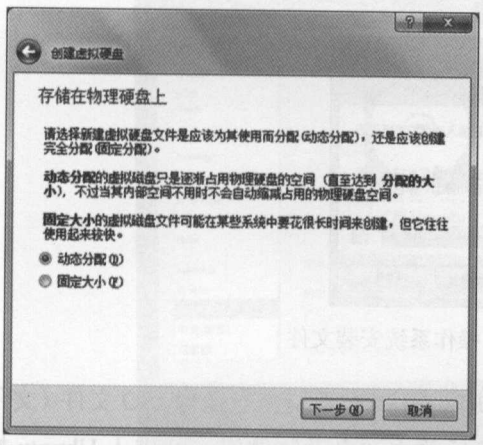


图 6-15　设置硬盘空间分配方式

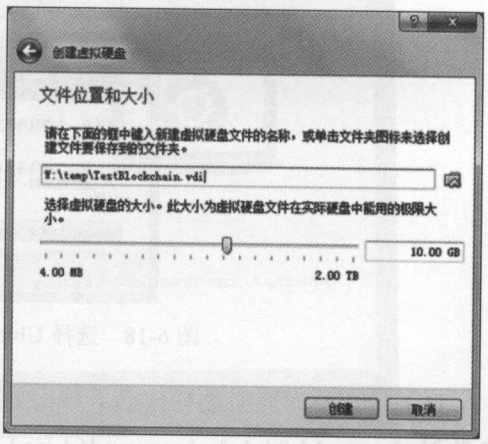


图 6-16　设置虚拟硬盘存放的位置和分配的大小

9）选择虚拟硬盘空间存储文件的存放位置和分配的大小（对于 Ubuntu 操作系统建议设置 10GB 以上），然后单击“创建”按钮。完成虚拟系统安装环境创建后，回到 VirtualBox 主窗口，如图 6-17 所示。

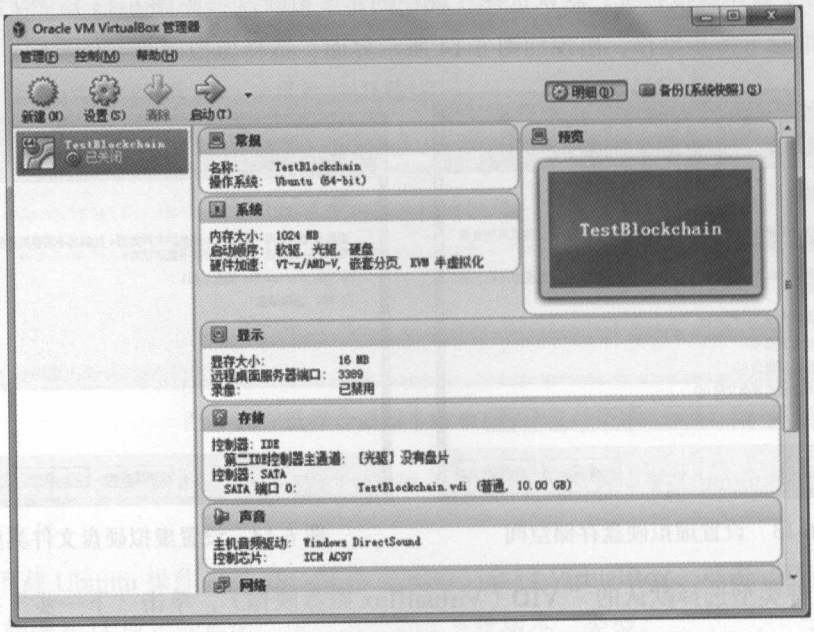


图 6-17　已创建的虚拟系统安装环境

10）从左侧列表中选择刚创建的虚拟系统，然后单击“启动”按钮继续下一步操作。第一次启动虚拟系统时的显示界面如图 6-18 所示。

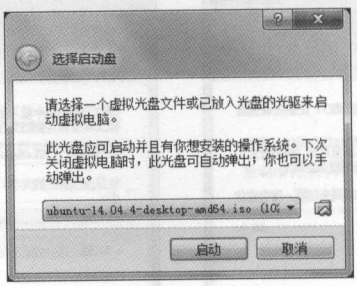


图 6-18　选择 Ubuntu 操作系统安装文件

11）从文件列表里选择你已下载完成的 Ubuntu 操作系统光盘镜像 ISO 文件（文件名类似 ubuntu-14.04.4-desktop-amd64.iso），然后单击“启动”按钮，就进入 Ubuntu 操作系统的安装初始界面了，如图 6-19 所示。

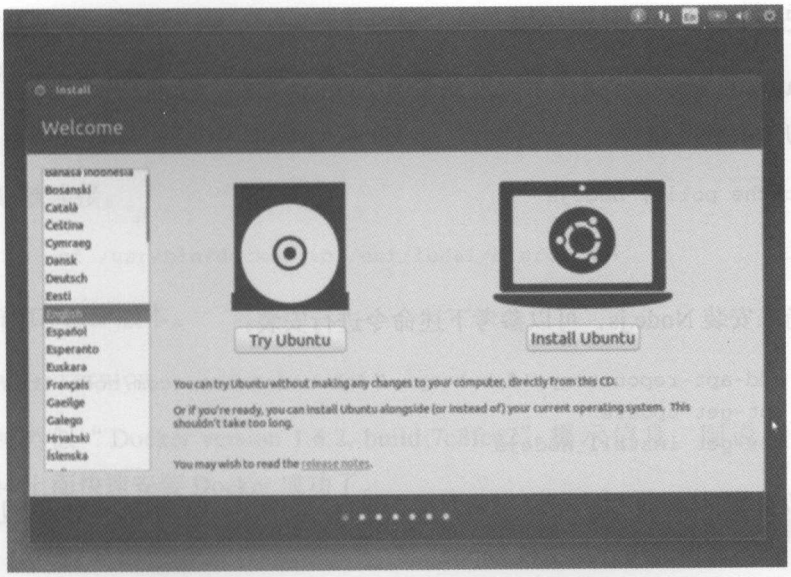


图 6-19　Ubuntu 操作系统安装初始界面

在这里，熟悉英文的读者直接点击 Install Ubuntu 按钮，然后根据后续各步骤中的提示安装就行了。对于不太熟悉英文的读者，可以从左侧语言列表中选择“中文简体”，安装界面将切换为中文提示，如图 6-20 所示。然后单击“安装 Ubuntu”按钮，根据后续各步骤按中文提示安装就行了。

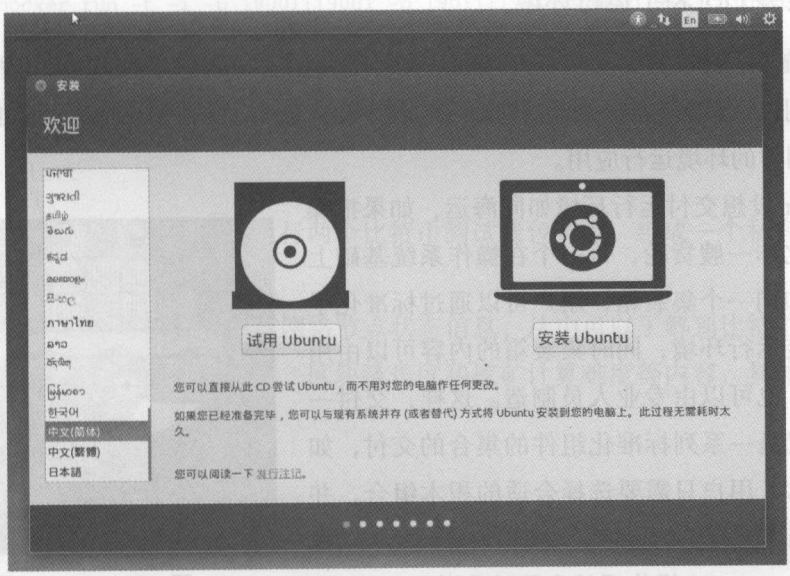


图 6-20　Ubuntu 操作系统中文安装界面

6.1.3 安装 Node.js 开发环境

Ubuntu14.04 Desktop 操作系统默认带有 Node.js 软件，查询是否已安装 Node.js 及相应版本的命令参考如下：

```
$apt-cache policy nodejs
$node -v
$npm -v
```

如果尚未安装 Node.js，可以参考下述命令进行安装：

```
$sudo add-apt-repository 'deb https://deb.nodesource.com/node trusty main'
$sudo apt-get update
$sudo apt-get install nodejs
```

确认已安装 Node.js 后，可以安装对应的 RPC 支持库。常用的 Node.js 的 RPC 支持库有多个，我们这里选用开源项目 kapitalize。

在 Ubuntu 终端命令行界面输入以下命令进行安装：

```
$npm install kapitalize
```

关于 kapitalize 开源项目的更多说明可以参考下述网址：

```
https://github.com/shamoons/Kapitalize
```

6.1.4 安装 Docker 运行环境

Docker（见图 6-21）是一个开源的应用容器（Container）引擎，让开发者可以打包他们的应用以及依赖包到一个可移植的容器镜像中，然后发布到支持容器的 Linux 机器上，可用同样的环境运行应用。

Docker 设想交付运行环境如同海运，如果把操作系统可比作一艘货轮，每一个在操作系统基础上的软件都如同一个集装箱，用户可以通过标准化手段自由组装运行环境，同时集装箱的内容可以由用户自定义，也可以由专业人员制造。这样，交付一个软件，就是一系列标准化组件的集合的交付，如同乐高积木，用户只需要选择合适的积木组合，并且在最顶端署上自己的名字。

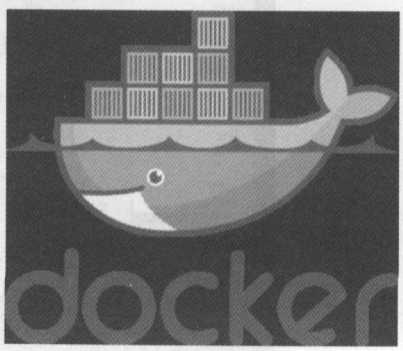


图 6-21 Docker 图标

在 Ubuntu14.04 操作系统上快速安装 Docker 运

行环境的方法如下。

1）在命令行下，输入下述命令安装 Docker 容器支持软件。

```
$sudo apt-get install docker.io
```

2）创建软连接。

```
$sudo ln -sf /usr/bin/docker.io /usr/local/bin/docker
```

3）查看 Docker 版本。

```
$docker --version
```

如看到类似“Docker version 1.6.2, build 7c8fca2”提示信息，则说明你已经在 Ubuntu14.04 上面快速安装 Docker 成功了。

如需了解关于 Docker 安装和运行的入门说明，可以通过网络搜索更多资料。

6.1.5　安装和运行比特币测试网络

1）下载比特币测试网络的 Docker 镜像。

```
$sudo docker pull freewil/bitcoin-testnet-box
```

2）运行 Docker 镜像。

```
$sudo docker run -t -i -p 19001:19001 -p 19011:19011 freewil/bitcoin-testnet-box
```

注意：上述命令中的 19001 和 19011 是配置给两个节点提供 RPC 服务的端口。

3）进入 Docker 运行环境后，输入下面的命令来启动比特币测试网络：

```
$ make start
```

启动成功后，将在本机模拟运行两个比特币测试钱包节点，组成一个私有范围的比特币测试网络。

输入下面的命令可以查看测试网络节点状态信息，从中可以了解到比特币测试网络的配置和运行状态，比如协议版本、区块链长度和挖矿计算难度等内容，具体见下面的详细说明。

```
$ make getinfo
```

显示的提示信息如下，中文为对其中内容的具体说明。

```
bitcoin-cli -datadir=1  getinfo // 第一个钱包节点的信息
```

```
{
    "version": 120100,                  //此客户端节点软件版本
    "protocolversion": 70012,           //比特币协议版本
    "walletversion": 60000,             //钱包数据格式版本
    "balance": 0.00000000,              //第一个钱包节点的账户余额，初始为0
    "blocks": 0,                        //已经产生的区块数量，初始启动为0，可以通过
                                        //进一步的命令来模拟生成区块数据
    "timeoffset": 0,                    //时间的时区偏移量
    "connections": 1,                   //本节点接入的其他节点数量
    "proxy": "",                        //网络代理设置
    "difficulty": 4.656542373906925e-10,    //当前挖矿计算难度
    "testnet": false,                   //是否使用外部的比特币测试网络，false为“否”，
                                        //即只建立私有测试环境
    "keypoololdest": 1467253951,//预生成的公匙和私匙池的起始时间
    "keypoolsize": 101,                 //预生成的公匙和私匙池的包含记录数量，用于生成钱包
                                        //地址和找零地址，这样钱包备份可以对已有的交易以及
                                        //未来多笔交易有效
    "paytxfee": 0.00000000,             //每次发送比特币的时候选择支付的交易手续费，包含
                                        //额外手续费的交易会更快地被包含在新生成的区块中，
                                        //因此会更快生效
    "relayfee": 0.00001000,             //每笔交易支付给矿工的最少的标准手续费
    "errors": ""                        //节点运行错误提示
}
bitcoin-cli -datadir=2  getinfo //第2个钱包节点的信息，具体说明同上
{
    "version": 120100,
    "protocolversion": 70012,
    "walletversion": 60000,
    "balance": 0.00000000,
    "blocks": 0,
    "timeoffset": 0,
    "connections": 1,
    "proxy": "",
    "difficulty": 4.656542373906925e-10,
    "testnet": false,
    "keypoololdest": 1467253951,
    "keypoolsize": 101,
    "paytxfee": 0.00000000,
    "relayfee": 0.00001000,
    "errors": ""
}
```

注意：以上运行参数一般用默认值即可，不需要调整。熟悉初步的区块链开发后，开发者如需调整以上运行参数，可以参考比特币客户端软件的配置说明。详见下述网址：https://en.bitcoin.it/wiki/Running_Bitcoin#Bitcoin.conf_Configuration_File。

4）初始化和测试区块链数据。

在 Docker 运行窗口里依次输入下面的命令来初始化创建基本的区块链数据，供进一步的程序示例来使用。

> **注意：在正式的比特币网络环境下，平均 10 分钟左右才能产生一个新的区块。但在这里特殊设定的测试网络（testnet）环境下，区块通过简单的命令控制就可以即时和批量产生，方便程序开发测试。**

5）模拟新产生 1 个区块记录。

```
make generate
```

6）模拟新产生 200 个区块记录。

```
make generate BLOCKS=200
```

7）查看最新的钱包状态，包括余额信息。这时可以留意看到第 1 个钱包节点的账户余额变为了 5050.00000000 BTC，即通过模拟区块挖矿产生的测试比特币。

```
make getinfo
```

8）给作为示例的测试钱包地址转账 10 个 BTC。

```
make sendfrom1 ADDRESS=mkiytxYA6kxUC8iTnzLPgMfCphnz91zRfZ AMOUNT=10
```

注意：这里的示例地址 mkiytxYA6kxUC8iTnzLPgMfCphnz91zRfZ 是比特币测试网络下的钱包地址（以字符 m 起始），与比特币正式网络下的钱包地址（一般以数字 1 或 3 起始）是有区别的。

9）模拟新产生 10 个区块记录，让上面的转账交易得到足够有效的确认。

```
make generate BLOCKS=10
```

10）查看最新的钱包状态包括余额信息，这时可以留意看到第 1 个钱包节点的账户余额变为了 5539.99996160 BTC，即已经交易转账支出了 10 个 BTC 加上少许的矿工费用。

```
make getinfo
```

6.1.6　运行第一个示例程序

下述示例代码以 Node.js 开发语言为例，演示了如何调用 RPC 接口来执行一些最基本的操作，比如导入比特币私钥，发送一个最简单的转账交易。开发者通过该示例可以

了解到最基本的区块链开发方法。

代码清单 6-1 示例程序 RpcTestnet.js 源码

```
//************************************************//
//Bitcoin-Testnet RPC sample of node.js           //
//        PPk Public Group © 2016.                //
//             http://ppkpub.org                  //
//     Released under the MIT License.            //
//************************************************//
//对应比特币测试网络(Bitcoin testnet)的RPC服务接口访问参数
var RPC_USERNAME='admin1';
var RPC_PASSWORD='123';
var RPC_HOST="127.0.0.1";
var RPC_PORT=19001;

//测试使用的钱包地址
TEST_ADDRESS='mkiytxYA6kxUC8iTnzLPgMfCphnz91zRfZ';
                                    //测试用的钱包地址，注意与比特币正式地址的区别
TEST_PRIVATE_KEY='cTAUfueRoL1HUXasWdnETANA7uRq33BUp3Sw88vKZpo9Hs8xWP82';
                                    //测试用的钱包私钥
TEST_WALLET_NAME='TestWallet1'; //测试的钱包名称

MIN_DUST_AMOUNT=10000;              //最小有效交易金额，单位为satoshi，即0.00000001 BTC
MIN_TRANSACTION_FEE=10000;          //矿工费用的最小金额，单位为satoshi

console.log('Hello, Bitcoin-Testnet RPC sample.');
console.log('   PPk Public Group © 2016   ');

//初始化访问RPC服务接口的对象
var client = require('kapitalize')()

client
    .auth(RPC_USERNAME, RPC_PASSWORD)
    .set('host', RPC_HOST)
    .set({
        port:RPC_PORT
    });

//显示当前连接的比特币测试网络信息
client.getInfo(function(err, info) {
    if (err) return console.log(err);
    console.log('Info:', info);
});

//查看当前钱包下属地址账户余额变动情况
```

```
client.listaccounts(function(err, account_list) {
    if (err) return console.log(err);
    console.log("Accounts list:\n", account_list);
});

//检查测试账号是否已存在于测试节点
client.getaccount(TEST_ADDRESS,function(err, result) {
    if (err || result!=TEST_WALLET_NAME ) { //如不存在，则新导入测试账号私钥
        console.log('Import the test account[',TEST_WALLET_NAME,']:',TEST_ADDRESS);
        client.importprivkey(TEST_PRIVATE_KEY,TEST_WALLET_NAME, function
            (err, imported_result) {
            if (err) return console.log(err);
            console.log('Imported OK:', imported_result);

            doSample();
        });
    }else{ //如已存在，则直接执行示例
        console.log('The test account[',TEST_WALLET_NAME,'] existed. Address:',
            TEST_ADDRESS);

        doSample();
    }

});

//示例实现功能
function doSample(){
    //获取未使用的交易 (UTXO) 用于构建新交易的输入数据块
    client.listunspent(6,9999999,[TEST_ADDRESS],function(err, array_unspent) {
        if (err) return console.log('ERROR[listunspent]:',err);
        console.log('Unspent:', array_unspent);

        var array_transaction_in=[];
        var sum_amount=0;
        for(var uu=0;uu<array_unspent.length;uu++){
            var unspent_record=array_unspent[uu];
            if(unspent_record.amount>0){
                sum_amount+=unspent_record.amount*100000000;
                    //注意：因为 JS 语言默认不支持 64 位整数，此处示例程序简单采用 32 位
                    //整数，能处理的交易涉及金额数值不大于 0xFFFFFFF (4294967295
                    //satoshi = 42.94967295 BTC)。实际应用程序需完善能处理 64 位
                    //整数功能
                array_transaction_in[array_transaction_in.length]={"txid":
                    unspent_record.txid,"vout":unspent_record.vout};
                if( sum_amount > (MIN_DUST_AMOUNT+MIN_TRANSACTION_FEE) )
```

```
                break;
            }
        }
    //确保新交易的输入金额满足最小交易条件
        if (sum_amount<MIN_DUST_AMOUNT+MIN_TRANSACTION_FEE) return console.
            log('Invalid unspent amount');
        console.log('Transaction_in:', array_transaction_in);
        //生成测试新交易的输出数据块，此处示例给指定目标测试钱包地址转账一小笔测试比特币
        //注意：输入总金额与给目标转账加找零金额间的差额为 MIN_TRANSACTION_FEE，就是
        //支付给比特币矿工的交易成本费用
        var obj_transaction_out={
            "mieC38pnPwMqbMAN6sGWwHRQ3msp7nRnNz":MIN_DUST_AMOUNT/100000000,
                //目标转账地址和金额
            "mkiytxYA6kxUC8iTnzLPgMfCphnz91zRfZ":(sum_amount-MIN_DUST_AMOUNT-
                MIN_TRANSACTION_FEE)/100000000 //找零地址和金额，默认用发送者地址
        };
        console.log('Transaction_out:', obj_transaction_out);
        //生成交易原始数据包
        client.createrawtransaction(array_transaction_in,obj_transaction_
            out,function(err2, rawtransaction) {
            if (err2) return console.log('ERROR[createrawtransaction]:',err2);
            console.log('Rawtransaction:', rawtransaction);
            //签名交易原始数据包
            client.signrawtransaction(rawtransaction,function(err3,
                signedtransaction) {
                if (err3) return console.log('ERROR[signrawtransaction]:',err3);
                console.log('Signedtransaction:', signedtransaction);
                var signedtransaction_hex_str=signedtransaction.hex;
                console.log('signedtransaction_hex_str:', signedtransaction_
                    hex_str);
                //广播已签名的交易数据包
                client.sendrawtransaction(signedtransaction_hex_str, false,
                    function(err4, sended) { //注意：第 2 个参数默认为 false，如
                    果设为 true 则指“Allow high fees to force it to spend”，
                    会强制将输入与输出金额差额部分作为支付给矿工的费用（谨慎！）
                    if (err4) return console.log('ERROR[sendrawtransaction]:',
                        err4);
                    console.log('Sended TX:', sended);
                        client.listaccounts(function(err, account_list) {
                        if (err) return console.log(err);
                        console.log("Accounts list:\n", account_list);
                            //发送新交易成功后，可以核对下账户余额变动情况
                    });
                });
            });
```

```
        });
    });
}
```

上述源码可以从下述网址下载，保存到测试环境下（保存文件名为 RpcTestnet.js）。

```
http://ppkpub.org/sample/RpcTestnet.js
```

然后在命令行下输入以下命令，即可运行并看到运行结果。

```
node RpcTestnet.js
```

注意：每运行一次测试代码后，都需要到 Docker 运行环境的命令行下输入 "make generate BLOCKS=10"，模拟产生新的区块记录，让测试代码产生的交易记录得到有效的确认。

在此程序的基础上，经过对 Bitcoin 协议的进一步了解，我们可以调用 RPC 接口进一步开发出更复杂功能，如自行构建特定交易数据包（比如备注信息、多重签名输出等）来满足特定业务需求，后文将继续深入介绍。

6.2　把握比特币“交易”数据结构

本节以比特币测试网络作为开发试验环境，解析比特币交易（Transaction）的数据结构，并以 Node.js 为例来说明如何自行组织特定需要的交易数据，并在签名后广播，最终被矿工节点确认生效。

6.2.1　了解比特币的“交易”数据结构

交易是比特币系统的信息载体和最小单元，而块（Block）就是将若干个这样的“交易”基础单元“打包装箱”，贴上“封条”，再按一定的机制和先后顺序将这些块串联起来，就构成了区块链（Blockchain）。

对于基于比特币区块链的应用开发，“交易”是最直接用到，也是最关键的数据结构。除了“交易”外，还需要掌握比特币区块链相关的一些基础术语的含义，包括钱包的私钥、公钥和地址、区块、区块链等，这些在本书前面的章节已有深入介绍。在本节我们侧重对“交易”的数据结构做更深入的剖析和了解。

一笔比特币交易是一个含有输入值和输出值的数据结构，该数据结构植入了将一笔

资金从初始点（输入值）转移至目标地址（输出值）的代码信息。一笔比特币交易包含的一些字段如表 6-1 所示。

表 6-1 比特币交易字段

字段名称	大 小	数据类型	描 述
协议版本	4 字节	uint32_t	明确这笔交易参照的规则协议的版本号
输入数量	1 ~ 9 字节	var_int	被包含的输入交易的数量
输入列表	不定	tx_in[]	一个或多个输入交易构成的数组
输出数量	1 ~ 9 字节	var_int	被包含的输出交易的数量
输出列表	不定	tx_out[]	一个或多个输出交易构成的数组
锁定时间	4 字节	uint32_t	一个 UNIX 时间戳或区块号

注意：锁定时间字段定义了能被加到区块链里的最早的交易时间。在大多数交易里，它被缺省设置成 0，用来表示立即执行。如果锁定时间大于 0 并且小于 5 亿，就被视为区块高度，意指在这个指定的区块高度之前，该交易不会被包含在区块链里。如果锁定时间大于 5 亿，则被当作是一个 UNIX 纪元时间戳（从 1970 年 1 月 1 日以来的秒数），并且在这个指定时点之前，该交易不会被包含在区块链里。锁定时间的使用相当于将一张纸质支票的生效时间予以后延。

基于区块链技术的应用开发，实际上主要就是在交易的输出数据结构上做文章，来承载具体的业务逻辑，比如 ODIN 开源项目就是将标识属性数据按一定格式嵌入比特币的多重交易输出数据块中。

6.2.2 交易记录的实例解析

下面是一个比特币交易的原始数据示例（将原始二进制数据按字节以十六进制形式输出，便于分析）。

```
0100000002eb2121e4e727bdb28525e79d39a90bd711b9e8413c054b29ffc4bb4775e69f82010000006b48
3045022100df82cf6c95b4eb64e4e9cee3af88a94c65fa81650e824d515f089192b7e3c09c0220119c1fcf
d9354755ea815cf714c181b56784b8f98f59f33e977c8939cd6f75db0121022e9f31292873eee495ca9744
fc410343ff373622cca60d3a4c926e58716114b9fffffffffc9f3b07ebfca68fd1a6339d0808fbb013c90c6
095fc93901ea77410103489ab7010000008a47304402206b993231adec55e6085e75f7dc5ca6c19e42e744
cd60abaff957b1c352b3ef9a022022a22fec37dfa2c646c78d9a0753d56cb4393e8d0b22dc580ef1aa6ccc
ef208d0141042ff65bd6b3ef04253225405ccc3ab2dd926ff2ee48aac210819698440f35d785ec3cec92a5
1330eb0c76cf49e9e474fb9159ab41653a9c1725c031449d31026affffffff0310270000000000000475121
```

```
022e9f31292873eee495ca9744fc410343ff373622cca60d3a4c926e58716114b9210250504b2d42455441
506565722d506565722d6e6574776f726b207075626c696352aee07b9a3b000000001976a914391ef5239d
a2a3904cda1fd995fb7c4377487ea988ac00000000000000000d6a0b436f6465206973204c617700000000
```

对上述报文按协议规则可按字段分解说明如下：

```
01000000          //版本号，UINT32

02                //Tx 输入数量，变长 INT。0x02=2 个输入

/*** 接下来是第 1 组 Input Tx ***/
eb2121e4e727bdb28525e79d39a90bd711b9e8413c054b29ffc4bb4775e69f82
                  //Tx 交易的 Hash 值，固定为 32 字节
01000000          //消费的 Tx 位于前向交易输出的第 0 个，UINT32，固定 4 字节
6b                //接下来对应签名数据的长度，0x6b = 107 字节
                  //这 107 字节长度的签名，含有两个部分：私钥签名 + 公钥
                  //当这里的数值为 00 时，则表示为尚未经过签名的原始交易
48                //对应私钥签名的数据长度，0x48 = 72 字节
3045022100df82cf6c95b4eb64e4e9cee3af88a94c65fa81650e824d515f089192b7e3c09c022011
9c1fcfd9354755ea815cf714c181b56784b8f98f59f33e977c8939cd6f75db01
                  //私钥签名内容
21                //对应公钥的数据长度，0x21 = 33 字节
022e9f31292873eee495ca9744fc410343ff373622cca60d3a4c926e58716114b9
                  //对应公钥数据
ffffffff          //序列号，UINT32，固定 4 字节。该字段是目前未被使用的交易替换功能，
                  //默认都设成 0xFFFFFFFF

/*** 第 2 组 Input Tx。与上同理，省略分解分析 ***/
c9f3b07ebfca68fd1a6339d0808fbb013c90c6095fc93901ea77410103489ab7010000008a473044
02206b993231adec55e6085e75f7dc5ca6c19e42e744cd60abaff957b1c352b3ef9a022022a22fec
37dfa2c646c78d9a0753d56cb4393e8d0b22dc580ef1aa6cccef208d0141042ff65bd6b3ef042532
25405ccc3ab2dd926ff2ee48aac210819698440f35d785ec3cec92a51330eb0c76cf49e9e474fb91
59ab41653a9c1725c031449d31026affffffff

03                //Tx 输出交易数量，变长 INT 类型。0x03=3 个输出

/*** 第 1 组输出 ***/
1027000000000000 //输出的比特币数量，UINT64，8 个字节。字节顺序需翻转得到
                  //0x0000000000002710 = 10000 satoshi = 0.0001 BTC
47                //输出描述脚本字节数，0x47 = 71 字节，由一些操作码与数值构成
51                //代表 OP_1（将脚本代码 1 压入堆栈）
21                //压入堆栈的第 1 个公钥的数据长度，0x21 = 33 字节
022e9f31292873eee495ca9744fc410343ff373622cca60d3a4c926e58716114b9
21                //压入堆栈的第 2 个公钥的数据长度，0x21 = 33 字节
0250504b2d42455441506565722d506565722d6e6574776f726b207075626c6963
52                //代表 OP_2 将脚本代码 2 压入堆栈。与前面的 0x51 合在一起表示 1of2 多重签名
```

```
ae                  //代表 OP_CHECKMULTISIG 执行多重签名验证

/*** 第 2 组输出 ***/
e07b9a3b00000000 //输出的比特币数量，UINT64，8 个字节。字节顺序需翻转
19                  //输出描述脚本字节数，0x19 = 25 字节，由一些操作码与数值构成
76                  //脚本起始操作，0x76 代表 OP_DUP 复制栈顶元素
a9                  //地址类型，0xa9 代表 OP_HASH160 栈顶项进行两次 HASH，先用 SHA256，
                    //再用 RIPEMD160
14                  //地址长度，0x14 = 20 字节
391ef5239da2a3904cda1fd995fb7c4377487ea9    //地址对应的 HASH160 值，20 字节
88                  //代表 OP_EQUALVERIFY 运行脚本的二进制算术和条件，如结果为 0，
                    //之后运行 OP_VERIFY
ac                  //代表 OP_CHECKSIG 交易所用的签名必须是哈希值和公钥的有效签名，
                    //如果为真，则返回 1


/*** 第 3 组输出 ***/
0000000000000000 //输出的比特币数量，UINT64，8 个字节。这是为了加入备注信息，
                    //不是普通转账交易，所以输出金额为 0
0d    //输出描述脚本字节数，0x0d = 13 字节，由一些操作码与数值构成
6a    //代表 OP_RETURN 标记交易无效，表示该交易只是追加的备注信息，不是普通转账交易
0b    //备注内容长度，0x0b = 11 字节
436f6465206973204c6177    //备注数据内容（将原始二进制数据按十六进制 ASCII 码形式表示）

00000000            //锁定时间，UINT32，固定 4 字节
```

通过上述解析，可以理解和掌握了比特币交易记录的常见组织格式，充分利用其中加注下划线的数据内容块来嵌入自定义的一些二进制数据内容，就可以实现自己特定的业务逻辑了。

6.2.3 运行示例程序

这里的示例程序是演示将一段特定内容的字符串按一定格式嵌入比特币交易的备注数据块中，这样就可以被存入比特币区块链上。

代码清单 6-2 示例程序 OpreturnTestnet.js 源码

```
//************************************************//
// RPC sample based Bitcoin-Testnet of node.js    //
//          PPk Public Group @2016.               //
//          http://ppkpub.org                     //
//      Released under the MIT License.           //
//************************************************//
//对应比特币测试网络（Bitcoin testnet）的 RPC 服务接口访问参数
```

```
var RPC_USERNAME='admin1';
var RPC_PASSWORD='123';
var RPC_HOST="127.0.0.1";
var RPC_PORT=19001;

//测试使用的钱包地址
TEST_ADDRESS='mkiytxYA6kxUC8iTnzLPgMfCphnz91zRfZ';
                                    //测试用的钱包地址，注意与比特币正式地址的区别
TEST_PUBKEY_HEX='022e9f31292873eee495ca9744fc410343ff373622cca60d3a4c926e5
    8716114b9';                     //十六进制表示的钱包公钥
TEST_HASH160='391ef5239da2a3904cda1fd995fb7c4377487ea9';
                                    //HASH160 格式的钱包公钥
TEST_PRIVATE_KEY='cTAUfueRoL1HUXasWdnETANA7uRq33BUp3Sw88vKZpo9Hs8xWP82';
                                    //测试用的钱包私钥
TEST_WALLET_NAME='TestWallet1'; //测试的钱包名称

MIN_DUST_AMOUNT=10000;              //最小有效交易金额，单位为 satoshi，即 0.00000001 BTC
MIN_TRANSACTION_FEE=10000;          //矿工费用的最小金额，单位为 satoshi

console.log('Hello, Bitcoin-Testnet RPC sample.');
console.log('      PPk Public Group @2016      ');

//初始化访问 RPC 服务接口的对象
var client = require('kapitalize')()

client
    .auth(RPC_USERNAME, RPC_PASSWORD)
    .set('host', RPC_HOST)
    .set({
        port:RPC_PORT
    });

//显示当前连接的比特币测试网络信息
client.getInfo(function(err, info) {
    if (err) return console.log(err);
    console.log('Info:', info);
});
//检查测试账号是否已存在于测试节点
client.getaccount(TEST_ADDRESS,function(err, result) {
    if (err || result!=TEST_WALLET_NAME ) { //如不存在，则新导入测试账号私钥
        console.log('Import the test account[',TEST_WALLET_NAME,']:',TEST_ADDRESS);
        client.importprivkey(TEST_PRIVATE_KEY,TEST_WALLET_NAME, function
            (err, imported_result) {
            if (err) return console.log(err);
```

```
            console.log('Imported OK:', imported_result);

            doRpcSample();
        });
    }else{ //如已存在，则直接执行示例
        console.log('The test account[',TEST_WALLET_NAME,'] existed. Address:',
            TEST_ADDRESS);

        doRpcSample();
    }

});

//示例实现功能
function doRpcSample(){
    //获取未使用的交易用于生成新交易
    client.listunspent(6,9999999,[TEST_ADDRESS],function(err2, array_unspent) {
        if (err2) return console.log('ERROR[listunspent]:',err2);
        console.log('Unspent:', array_unspent);

        //测试数据定义
        var TEST_DATA='Peer-Peer-network is the future!';
        console.log('TEST_DATA=',TEST_DATA);

        //将原始字节字符串转换为用十六进制表示
        var str_demo_hex=stringToHex(TEST_DATA);
        console.log('str_demo_hex=',str_demo_hex);

        //生成输入交易定义块
        var min_unspent_amount=MIN_DUST_AMOUNT*1+MIN_TRANSACTION_FEE;
        var array_transaction_in=[];

        var sum_amount=0;
        for(var uu=0;uu<array_unspent.length;uu++){
            var unspent_record=array_unspent[uu];
            if(unspent_record.amount>0){
                sum_amount+=unspent_record.amount*100000000;
                array_transaction_in[array_transaction_in.length]={"txid":
                    unspent_record.txid,"vout":unspent_record.vout};

                if( sum_amount > min_unspent_amount )
                    break;
            }
        }
        //确保新交易的输入金额满足最小交易条件
```

```
    if (sum_amount<=min_unspent_amount) return console.log
        ('Invalid unspent amount');
console.log('Transaction_in:', array_transaction_in);
//构建原始交易数据
var rawtransaction_hex = '01000000';   //Bitcoin 协议版本号, UINT32
rawtransaction_hex += byteToHex(array_transaction_in.length) ;
                                       //设置输入交易数量
for(var kk=0;kk<array_transaction_in.length;kk++){
    rawtransaction_hex += reverseHex(array_transaction_in[kk].txid)+
        uIntToHex(array_transaction_in[kk].vout);
    rawtransaction_hex += "00ffffffff";//签名数据块的长度和序列号,
                                       //00 表示尚未签名
}
rawtransaction_hex += byteToHex(2);    //设置输出交易数量
//使用 op_return 对应的备注脚本空间来嵌入自定义数据
rawtransaction_hex += "0000000000000000";
rawtransaction_hex += byteToHex(2+str_demo_hex.length/2) + "6a" +
    byteToHex(str_demo_hex.length/2) +str_demo_hex;
//最后添加一个找零输出交易
var charge_amount = sum_amount - MIN_TRANSACTION_FEE;
console.log('sum_amount:', sum_amount);
console.log('min_unspent_amount:', min_unspent_amount);
console.log('charge_amount:', charge_amount);
console.log('uIntToHex(',charge_amount,')=', uIntToHex(charge_amount));

rawtransaction_hex += uIntToHex(charge_amount)+"00000000";
                                       //找零金额,UINT64
rawtransaction_hex += "1976a914" + TEST_HASH160 +"88ac";
                                       //找零地址为发送者的钱包地址
rawtransaction_hex += "00000000";      //锁定时间,默认设置成 0,表示立即
                                       //执行,这是整个交易数据块的结束字段
console.log('Rawtransaction:', rawtransaction_hex);
//签名交易原始数据包
client.signrawtransaction(rawtransaction_hex,function(err3,
    signedtransaction) {
    if (err3) return console.log('ERROR[signrawtransaction]:',err3);
    console.log('Signedtransaction:', signedtransaction);
    if (!signedtransaction.complete) return console.log
        ('signrawtransaction failed');
    var signedtransaction_hex_str=signedtransaction.hex;
    console.log('signedtransaction_hex_str:', signedtransaction_hex_str);
    //广播已签名的交易数据包
    client.sendrawtransaction(signedtransaction_hex_str, false,
        function(err4, sended){
        //注意第 2 个参数默认为 false,如果设为 true 则指 Allow high fees to
```

```
                //force it to spend
                //会强制发送交易，并将输入与输出金额差额部分作为矿工费用（谨慎！）
                if (err4) return console.log('ERROR[sendrawtransaction]:',err4);
                console.log('Sended TX:', sended);
            });
        });
    });
}

//1字节整数转换成十六进制字符串
function byteToHex(val){
    var resultStr='';
    var tmpstr=parseInt(val%256).toString(16);
    resultStr += tmpstr.length==1? '0'+tmpstr : tmpstr;

    return resultStr;
}

//将HEX字符串反序输出
function reverseHex(old){
    var array_splited=old.match(/.{2}|.+$/g);
    var reversed='';
    for(var kk=array_splited.length-1;kk>=0;kk--){
        reversed += array_splited[kk];
    }
    return reversed;
}
//32位无符号整数变成十六进制数，并按翻转字节顺序
function uIntToHex(val){
    var resultStr='';
    var tmpstr=parseInt(val%256).toString(16);
    resultStr += tmpstr.length==1? '0'+tmpstr : tmpstr;

    tmpstr=parseInt((val%65536)/256).toString(16);
    resultStr += tmpstr.length==1? '0'+tmpstr : tmpstr;

    tmpstr=parseInt(parseInt(val/65536)%256).toString(16);
    resultStr += tmpstr.length==1? '0'+tmpstr : tmpstr;

    tmpstr=parseInt(parseInt(val/65536)/256).toString(16);
    resultStr += tmpstr.length==1? '0'+tmpstr : tmpstr;

    return resultStr;
}

//将Ascii或Unicode字符串转换成十六进制表示
```

```
function stringToHex(str){
    var val="";
    for(var i = 0; i < str.length; i++){
        var tmpstr=str.charCodeAt(i).toString(16);  //Unicode
        val += tmpstr.length==1? '0'+tmpstr : tmpstr;
    }
    return val;
}
```

上述源码可以从下述网址下载，并保存到测试环境下（保存文件名为 Opreturn-Testnet.js）。

```
http://ppkpub.org/sample/OpreturnTestnet.js
```

然后在命令行下输入以下命令，即可运行并看到运行结果。

```
node OpreturnTestnet.js
```

6.3　实战：多重签名交易

本节以比特币测试网络作为开发试验环境，结合 Node.js 实现开源项目 ODIN 的标识注册功能，说明如何利用多重签名交易形式来嵌入自定义数据，签名广播后，再被矿工节点确认存入区块链后，最终能被读取、解析，得到注册结果。这样就实现了一个从写入区块链到从区块链读取的完整过程。

6.3.1　将 ODIN 标识注册到区块链上的实例解析

ODIN 是 Open Data Index Name 即“开放数据索引命名标识”的缩写，第 10 章中会介绍 ODIN 更多的技术细节。广义上说，ODIN 是指在网络环境下标识和交换数据内容索引的一种开放式系统。ODIN 的实现关键是把数据嵌入比特币交易的多重签名输出数据块中，对于 1-of-N 输出，每个数据块的第 1 个公钥固定是发送者的，第 2 ~ N 个公钥的地址空间用来存放编码的 ODIN 消息数据。下面一段文字是 ODIN 技术规范里对于“新注册 ODIN 标识”的具体消息定义。

其中：

- 比特币源地址对应 ODIN 标识注册者。
- 比特币目的地址对应 ODIN 标识拥有者。

消息数据块的格式按字节顺序定义如下：

第 1 ~ 32 字节：前缀特征标识，32 个字节的 ASCII 字符串 "P2P is future! ppkpub.org->ppk:0"（不含双引号）
第 33 字节：消息类型，1 个字节，取值为 ASCII 字符 R
第 34 字节：消息正文数据格式，1 个字节
取值定义：ASCII 字符。
T 表示“UTF-8 编码文本字符串”，
G 表示“经 gzip 算法压缩得到的二进制数据，需解压后可得到 UTF-8 编码的原始文本字符串”
第 35 ~ 36 字节：消息正文数据字节长度，2 个字节的无符号短整型二进制数据，取值为 0-65535
第 37 字节到消息正文指定长度结束，是按字节存放的消息正文数据，需根据第 34 字节的数据格式取值来获得原始消息文本，为 UTF-8 编码的 JSON 格式字符串，对应一个 JSON 对象数据，说明如下：

```
{
    "title":" 说明：个体名称字符串 ",
    "email":" 说明：个体的公开 EMAIL，可选 ",
    "auth":" 说明：配置权限，取值定义见下方注释 ",
    "ap_list":[" 说明：若干个数据访问点 AP 的 URL 数组，最少需填写一个 ",...,"xxxx"],
    "catalog":" 说明：数据源类型，可选保留字段，待补充 "
}
```

配置权限的取值说明：ASCII 字符 0、1 或 2。

- 0 表示注册者或拥有者任一方都可以修改拥有者相关信息。
- 1 表示只有注册者能修改拥有者相关信息。
- 2 表示注册者和拥有者必须共同确认才能修改拥有者相关信息。

假设有下述作为示例的一段 ODIN 标识注册信息：

```
{"title":"PPk-ODIN-sample","email":"ppkpub@gmail.com","auth":"2","ap_list":
    ["http://ppkpub.org/AP/"]}
```

那么就可以按照上述的消息定义，将其组装为一条比特币交易记录，并广播到比特币网络上生效。对应交易的原始数据示例如下（将原始二进制数据按字节以十六进制 ASCII 码形式输出，便于分析）。

```
01000000032237b858f1a697cc2d26a451bd3fd3ef1944eb53f579b4fac38e5ecb5c0fc42c010000006b48
3045022100da55a2d9f97695db12aecc0113662437957a6d4f17064ff49602ddc39904c31302201e81eae0
c84f25019485ae4a2ce9b67c0e8485599df87ab876b469e3cbbd24100121022e9f31292873eee495ca9744
fc410343ff373622cca60d3a4c926e58716114b9ffffffff2ef89686bebf72bd31b8f27780223f7b5f448d
0110b6fdda19595a073f42a301000000006b483045022100d49360fa6bd45b92a068127db31c9cfd93af87
543799968a5b076c2fea151f9b0220647900f5fc763f5a3eed13d382e13a3bddd15646867b56f1be9d2629
b2ccb4360121022e9f31292873eee495ca9744fc410343ff373622cca60d3a4c926e58716114b9ffffffff
d704b1c1977cd50be182134b18fafaa16db1e917dfe4f93bcab1584aabf323d4010000006b483045022100
fb88f75cae8accfe969cd89afbca677ff78a4914f5f506d6e5d481baf484e9f2022039f560414ec5a778a1
9565f7fe9e51b6acf7b841b4ba2188785a5bb6051d7d510121022e9f31292873eee495ca9744fc410343ff
373622cca60d3a4c926e58716114b9ffffffff037d160000000000001976a91451a09d25106715f09a14ca
c6367c3f4f2408590d88ac7d160000000000000cf5121022e9f31292873eee495ca9744fc410343ff373622
```

```
cca60d3a4c926e58716114b9212050325020697320667574757265212070706b7075622e6f72672d3e7070
6b3a302120525400657b227469746c65223a2250506b2d4f44494e2d73616d706c65222c222120656d6169
6c223a2270706b70756240676d61696c2e636f6d222c22617574682221203a2232222c2261705f6c697374
223a5b22687474703a2f2f70706b7075622e6f210972672f41502f225d7d000000000000000000000000000
000000000000000000000056aee6cac223000000001976a914391ef5239da2a3904cda1fd995fb7c4377487e
a988ac00000000
```

上述报文按协议规则可按字段分解说明如下：

```
01000000            //版本号，UINT32
03                  //Tx 输入数量，变长 INT。0x03=3 个输入
/*** 接下来是第 1 组 Input Tx ***/
2237b858f1a697cc2d26a451bd3fd3ef1944eb53f579b4fac38e5ecb5c0fc42c
                    //Tx 交易的 Hash 值，固定 32 字节
01000000            //消费的 Tx 位于前向交易输出的第 0 个，UINT32，固定 4 字节
6b                  //接下来对应签名数据的长度，0x6b = 107 字节
                    //这 107 字节长度的签名，含有两个部分：私钥签名 + 公钥
                    //当这里的数值为 00 时，则表示为尚未经过签名的原始交易
48                  //对应私钥签名的数据长度，0x48 = 72 字节
3045022100da55a2d9f97695db12aecc0113662437957a6d4f17064ff49602ddc39904c31302201e
81eae0c84f25019485ae4a2ce9b67c0e8485599df87ab876b469e3cbbd241001
                    //私钥签名内容
21                  //对应公钥的数据长度，0x21 = 33 字节
022e9f31292873eee495ca9744fc410343ff373622cca60d3a4c926e58716114b9
                    //对应公钥数据
ffffffff            //序列号，UINT32，固定 4 字节。该字段是目前未被使用的交易替换功能，
                    //默认都设成 0xFFFFFFFF
/*** 第 2 组 Input Tx。与上同理，省略分解 ***/
2ef89686bebf72bd31b8f27780223f7b5f448d0110b6fdda19595a073f42a301000000006b483045
022100d49360fa6bd45b92a068127db31c9cfd93af87543799968a5b076c2fea151f9b0220647900
f5fc763f5a3eed13d382e13a3bddd15646867b56f1be9d2629b2ccb4360121022e9f31292873eee4
95ca9744fc410343ff373622cca60d3a4c926e58716114b9ffffffff
/*** 第 3 组 Input Tx。与上同理，省略分解 ***/
d704b1c1977cd50be182134b18fafaa16db1e917dfe4f93bcab1584aabf323d4010000006b483045
022100fb88f75cae8accfe969cd89afbca677ff78a4914f5f506d6e5d481baf484e9f2022039f560
414ec5a778a19565f7fe9e51b6acf7b841b4ba2188785a5bb6051d7d510121022e9f31292873eee4
95ca9744fc410343ff373622cca60d3a4c926e58716114b9ffffffff
03                  //Tx 输出交易数量，变长 INT 类型。0x03=3 个输出
/*** 第 1 组输出 ***/
7d16000000000000 //输出的比特币数量，UINT64，8 个字节。字节顺序需翻转得到
                    //0x000000000000167d = 5757 satoshi = 0.00005757 BTC
19                  //输出描述脚本字节数，0x19 = 25 字节，由一些操作码与数值构成
76                  //脚本起始操作，0x76 代表 OP_DUP 复制栈顶元素
a9                  //地址类型，0xa9 代表 OP_HASH160，即栈顶项进行两次 HASH，先用
                    //SHA256，再用 RIPEMD-160
14                  //地址长度，0x14 = 20 字节
```

```
51a09d25106715f09a14cac6367c3f4f2408590d
                    // 对应 ODIN 标识拥有者地址的 HASH160 值，20 字节
88                  // 代表 OP_EQUALVERIFY 运行脚本的二进制算术和条件，如果结果为 0，
                    // 之后运行 OP_VERIFY
ac                  // 代表 OP_CHECKSIG 交易所用的签名必须是哈希值和公钥的有效签名，
                    // 如果为真，则返回 1
/*** 第 2 组输出 ***/
7d16000000000000 // 输出的比特币数量，UINT64，8 个字节。字节顺序需翻转
cf                  // 输出描述脚本字节数，0xcf = 207 字节，由一些操作码与数值构成
51                  // 代表 OP_1 将脚本代码 1 压入堆栈
21                  // 压入堆栈的第 1 个公钥的数据长度，0x21 = 33 字节。对应 ODIN 标识
                    // 注册者地址的公钥
022e9f31292873eee495ca9744fc410343ff373622cca60d3a4c926e58716114b9
21                  // 压入堆栈的第 2 个公钥的数据长度。从第 2 个公钥开始嵌入 ODIN 标识消息内容
20503250206973206675747572652120707 06b7075622e6f72672d3e70706b3a30
21                  // 压入堆栈的第 3 个公钥的数据长度
20525400657b227469746c65223a2250506b2d4f44494e2d73616d706c65222c22
21                  // 压入堆栈的第 4 个公钥的数据长度
20656d61696c223a2270706b70756240676d61696c2e636f6d222c226175746822
21                  // 压入堆栈的第 5 个公钥的数据长度
203a2232222c2261705f6c697374223a5b22687474703a2f2f70706b7075622e6f
21                  // 压入堆栈的第 6 个公钥的数据长度
0972672f41502f225d7d0000000000000000000000000000000000000000000000
56                  // 代表 OP_6 将脚本代码 6 压入堆栈。与前面的 0x51 组合在一起表示 1of6 多重签名
ae                  // 代表 OP_CHECKMULTISIG 执行多重签名验证
/*** 第 3 输出 ***/
e6cac22300000000 // 输出的比特币数量，UINT64，8 个字节。字节顺序需翻转
19                  // 输出描述脚本字节数，0x19 = 25 字节，由一些操作码与数值构成
76                  // 脚本起始操作，0x76 代表 OP_DUP 复制栈顶元素
a9                  // 地址类型，0xa9 代表 OP_HASH160，即栈顶项进行两次 HASH，先用 SHA256，
                    // 再用 RIPEMD-160
14                  // 地址长度，0x14 = 20 字节
391ef5239da2a3904cda1fd995fb7c4377487ea9    // 对应的 HASH160 值，20 字节
88                  // 代表 OP_EQUALVERIFY 运行脚本的二进制算术和条件，如果结果为 0，
                    // 之后运行 OP_VERIFY
ac                  // 代表 OP_CHECKSIG 交易所用的签名必须是哈希值和公钥的有效签名，
                    // 如果为真，则返回 1
00000000            // 锁定时间，UINT32，固定 4 字节
```

通过上述解析，结合前文所说明的 ODIN 协议定义内容，以及上述交易数据中的加注下划线的内容，就可以还原解析出对应的 ODIN 标识注册消息。

6.3.2 运行示例程序

示例程序包括两部分。

（1）OdinMonitorTestnet.js

监测比特币测试网络 Testnet 相关区块链数据的变化，从中解析出新注册的 ODIN 标识。

源码可以从下述网址下载：http://ppkpub.org/sample/OdinMonitorTestnet.js。

（2）OdinRegisterTestnet.js

在比特币测试网络 Testnet 注册一个新的 ODIN 标识。

源码可以从下述网址下载：http://ppkpub.org/sample/OdinRegisterTestnet.js。

将上述示例代码下载并保存到测试环境下（保存文件名为 OdinRegisterTestnet.js 和 OdinMonitorTestnet.js）。在开发测试环境的命令行下输入以下命令，启动监测示例程序。

```
node OdinMonitorTestnet.js
```

然后新开启一个文本终端窗口，在命令行下输入以下命令运行注册示例程序。

```
node OdinRegisterTestnet.js
```

运行上述注册示例程序后，到比特币测试网络的 Docker 运行环境的命令行下输入“make generate BLOCKS=10”，模拟产生新的区块记录，刚产生的交易记录就会得到有效的确认，并被存入区块链中。这时在监测程序的运行界面上就会提示解析到新的 ODIN 标识注册记录，如图 6-22 所示。

```
Found a new ODIN[ 218.1 ]:
 { register_pubkey: '022e9f31292873eee495ca9744fc410343ff373622cca60d3a4c926e587
16114b9',
  owner_address_hash160: '391ef5239da2a3904cda1fd995fb7c4377487ea9',
  setting:
   { title: 'PPk-ODIN-sample',
     email: 'ppkpub@gmail.com',
     auth: '2',
     ap_list: [ 'http://ppkpub.org/AP/' ] } }
```

图 6-22　监测程序显示示例

6.4　本章小结

刚接触比特币及其底层区块链技术的开发者常常会面临问题：如何上手比特币区块链应用开发技术？如何了解它的关键技术实践点？本章用简短的篇幅，结合笔者在 PPk ODIN 开源项目中的实践经验，用具体案例来阐述和介绍相关开发知识。读者通过学习本章，可以理解 RPC 协议接口和关键的比特币“交易”数据结构，进而举一反三，就可以灵活应用到自己的开发实践中。

第7章

智能合约

在第 1 章中，我们首先通过一个例子将读者引入区块链的世界，之后详尽地介绍了区块链的背景、基础知识以及构架，并深层次地分析了区块链背后的技术以及所遇到的问题。在这一章中，我们将介绍在未来区块链技术发展中最重要的应用场景，即智能合约的实现。

在第 2 章中，我们曾经介绍过一个重要的区块链应用平台——以太坊。众所周知，在以太坊平台上，最重要的应用就是设计部署智能合约。那什么是智能合约？智能合约能做什么？如何在以太坊上部署智能合约呢？在这一章中，我们将结合具体的案例逐一解答这些问题。

7.1 智能合约简介

7.1.1 什么是智能合约

虽然在法律范畴上来说，智能合约是否是一个真正意义上的合约还有待研究确认，但在计算机科学领域，智能合约是指一种计算机协议，这类协议一旦制定和部署就能实现自我执行（self-executing）和自我验证（self-verifying），而且不再需要人为的干预。从技术角度来说，智能合约可以被看作一种计算机程序，这种程序可以自主地执行全部或部分和合约相关的操作，并产生相应的可以被验证的证据，来说明执行合约操作的有效性。在部署智能合约之前，与合约相关的所有条款的逻辑流程就已经被制定好了。智能

合约通常具有一个用户接口（interface），以供用户与已制定的合约进行交互，这些交互行为都严格遵守此前制定的逻辑。得益于密码学技术，这些交互行为能够被严格地验证，以确保合约能够按照此前制定的规则顺利执行，从而防止出现违约行为。

举个例子来说，对银行账户的管理就可以看成一组智能合约的应用。在传统方式中，对账户内存款的操作需要中心化的银行进行授权，离开银行的监管，用户就连最简单的存取款都无法进行。智能合约能够完全代替中心化的银行职能，所有账户操作都可以预先通过严密的逻辑运算制定好，在操作执行时，并不需要银行的参与，只要正确地调用合约即可。再比如说，用户的信息登记系统完全可以由智能合约实现，从而完全抛开需要人为维护的中心化数据管理方式，用户可以通过预先定义好的合约实现信息登记、修改、注销等功能。此外，通过设计更复杂的合约，智能合约几乎可以应用于任何需要记录信息状态的场合，例如各种信息记录系统以及金融衍生服务。但这要求合约设计者能够深入了解流程的各个细节，并进行合理设计，因为通常来说，智能合约一旦部署成功，就不会再受到人为的干预，从而无法随时修正合约设计中出现的漏洞。

7.1.2 智能合约的历史

在20世纪七八十年代，随着计算机的发明，对计算机的理论研究达到了一个高潮。研究人员致力于让计算机帮助人类从事更多的工作，从而解放人类的生产劳动。正是在此时，人们提出了让计算机代替人类进行商业市场管理的想法。与此同时，公钥密码学得到革命性的发展，但使计算机完全代替人类进行商业管理的技术并未成熟。

直到20世纪90年代，从事数字合约和数字货币研究的计算机科学家尼克萨博（Nick Szabo）第一次提出了“智能合约”这一说法，其致力于将已有的合约法律法规以及相关的商业实践转移到互联网上来，使得陌生人通过互联网就可以实现以前只能在线下进行的商业活动，并实现真正的完全的电子商务。1994年，尼克萨博对智能合约做出以下描述[1]：

“智能合约是一个由计算机处理的、可执行合约条款的交易协议。其总体目标是能够满足普通的合约条件，例如支付、抵押、保密甚至强制执行，并最小化恶意或意外事件发生的可能性，以及最小化对信任中介的需求。智能合约所要达到的相关经济目标包括降低合约欺诈所造成的损失，降低仲裁和强制执行所产生的成本以及其他交易成本等。”

尼克萨博以及其他研究者希望借助密码学协议以及其他数字化安全机制，实现逻辑清楚、检验容易、责任明确和追责简单的合约，这将极大地改进传统的合约制定和履行方式，并降低相关的成本，将所有的合约条款以及操作置于计算机协议的掌控之下。但

那时，很多技术还不成熟，并无法完全实现研究者的想法，这一局面在比特币的出现之后得到很大的改观。借由比特币背后的区块链技术，智能合约得以飞速发展，有许多研究机构已将区块链上的智能合约作为未来互联网合约的重要研究方向，很多智能合约项目已经初步得以实现，并吸引大量的资金投入其中。

7.1.3 智能合约的优点和面临的风险

现今，虽然智能合约还未被广泛应用和实践，但其优点已得到研究人员和业内人士的广泛认可。总体来说，智能合约具有以下优点：

1）**高效的实时更新**：由于智能合约的执行不需要人为的第三方权威或中心化代理服务的参与，其能够在任何时候响应用户的请求，大大提升了交易进行的效率。用户不需要等待银行开门就可以办理相关的业务，只要通过网络一切都可以方便快捷地解决。

2）**准确执行**：智能合约的所有条款和执行过程是提前制定好的，并在计算机的绝对控制下进行。因此所有执行的结果都是准确无误的，不会出现不可预料的结果。这也是传统合约制定和执行过程中所期望的。现今，智能合约的准确执行得益于密码学的发展和区块链技术的发明。

3）**较低的人为干预风险**：在智能合约部署之后，合约的所有内容都将无法修改，合约中的任何一方都不能干预合约的执行，也就是说任何合约人都不能为了自己的利益恶意毁约，即使发生毁约事件，事件的责任人也会受到相应的处罚，这种处罚也是在合约制定之初就已经决定好的，在合约生效之后无法更改。

4）**去中心化权威**：一般来说，智能合约不需要中心化的权威来仲裁合约是否按规定执行，合约的监督和仲裁都由计算机来完成。在区块链上的智能合约更具有这一特性，在一个区块链网络中一般不存在一个绝对的权威来监督合约的执行，而是由该网络中绝大部分的用户来判断合约是否按规定执行，这种大多数人监督的方式是由 PoW 或 PoS 技术来实现的。如果将这种情况搬到现实世界中，或许现在的所有法官都要失业了，而与此同时我们每个人都是法官，都参与监督和仲裁。

5）**较低的运行成本**：正因为智能合约具有去人为干预的特点，其能够大大减少合约履行、裁决和强制执行所产生的人力成本，但要求合约制定人能够将合约的各个细节在合约建立之初就确定下来。这可能会使在传统行业（如银行）工作的部分员工面临失业，但从长远来说会促进行业的转型，向更新更好的领域发展。

虽然智能合约具有许多显而易见的优点，但对智能合约的深入研究才刚刚开始，其广泛应用还面临着潜在的甚至是毁灭性的各类风险。其中一个已知的风险恰恰是来自于

智能合约的去人为干预的特性。

2016年4月，史上最大的一个众筹项目The DAO正式上线。经过一个多月的众筹，总共募集到超过价值1.5亿美元的以太币用于建立该项目。从这令人震惊的数字上可以看出区块链技术以及之后的智能合约广泛应用的前景是多么让人充满信心。但就在短短一个多月之后，The DAO所在的平台以太坊的创始人之一Vitalik Buterin在其Slock.it社区里面发表声明，表示The DAO存在巨大的漏洞，在其上的大量的以太币已经被"偷"，未来或许还会有大量的以太币被偷，而The DAO的设计执行者对此攻击却无能为力。这一攻击的出现，恰恰是因为The DAO的智能合约在设计之初就存在漏洞，由于基于区块的智能合约的去人为干预特性，这一漏洞无法被线上修复，只能眼睁睁地看着黑客把更多的以太币从项目中偷走。虽然在后续的对策研究中，以太坊的设计者们想出了让以太坊分叉的解决办法来挽回损失（从根本上将丢失以太币的交易作废），但很多分叉的反对者认为，人为分叉完全背离了去中心化思想，并会大大降低以太坊在人们心目中的信用。由于分歧的存在，人们发起了投票，以决定是否分叉。无论最终是否分叉，都将会对The DAO以及未来的智能合约发展产生深远的影响，迫使合约的设计者将工作重点放到讨论合约的安全性上来。此外，由于智能合约具有自我验证的特性，其上的数据隐私保护也面临着巨大的风险。

The DAO攻击事件的发生恰恰是由于其公认的优点，这很值得业内人士反思，技术的应用要有坚实的理论基础做支撑，那么完全去中心化的智能合约是否已经成熟以及面临攻击该如何应对都将成为未来主要探讨的课题。但不管怎样，业内人员普遍认为，区块链技术和智能合约都将成为未来互联网发展的重要方向，现在面临的挫折是新技术成熟的必然过程。

我们在第9章会详细介绍The DAO事件的来龙去脉。

7.2 以太坊智能合约详解

这一节我们将结合最前沿的智能合约平台——以太坊，进一步介绍其上的智能合约。阅读这一节的读者需要对区块链技术和智能合约有一定的了解，如需补充基础知识，请阅读之前的几章。

7.2.1 以太坊上的账户

账户是以太坊的核心操作对象，但和比特币以及传统的区块链不同，在以太坊上，

账户被分为两类：一类叫作外部所有账户（Externally Owned Accounts，EOA），另一类叫作合约账户。

其中外部所有账户可被简单称为“账户”，这是由于外部所有账户与一般的区块链电子货币的账户（例如，比特币账户）类似，都是人为创建的、能够存取货币、由公钥加密系统加密和分享的账户。不同的是，在以太坊上，外部所有账户有能力创建合约账户，并部署智能合约。总之，在以太坊内部，外部所有账户和合约账户都被统称为状态对象（state objects），这些对象都具有自己的状态，其中外部所有账户的状态体现在其包含多少货币余额，而合约账户既含有货币余额状态还有合约存储状态。这些对象的状态随着每个区块的产生而发生变化（也可能不变），因此，简而言之，以太坊的基础就是通过区块链技术记录这些状态的变化，通过工作量证明，这些状态变化被大多数用户所认可，从而达成共识。

1. 钥匙文件

每一个账户都通过一对私钥和公钥来确定。每个账户都拥有一个地址，这个地址来自于该账户的公钥的最后 20 个字节。每一账户的地址和私钥都被编码成一个 JSON 格式的“钥匙文件”（keyfile）。因此，以太坊用户不能通过文本编辑器直接看到自己的私钥。存储在本地的以太坊账户私钥总是处于加密状态，而加密所使用的密钥就是在账户创建时用户所输入的密码。这一机制是为了保护账户的安全。以下就是一个钥匙文件所存储的数据。

```
{
    "address": "24265935827a9332a97cd0db938f2e0e0855853c",
    "Crypto": {
        "cipher": "aes-128-ctr",
        "ciphertext": "eb59eebe4b944627939db96db8e0d7125d85495965a41e29a0c
            102465a0898e5",
        "cipherparams": {
            "iv": "20391682b7c25dcf29e1b8c48312e4cc"
        },
        "kdf": "scrypt",
        "kdfparams": {
            "dklen": 32,
            "n": 262144,
            "p": 1,
            "r": 8,
            "salt": "3932589c8ce28c0d1c15e5e888c45a73145e6d141f729ee5e7196
                8086636e335"
        },
```

```
        "mac": "08a95f73166a70ef5b1846196d6af170781721a4972440ce41d05ba3f44fcc5b"
    },
    "id": "76aa7f19-cd6a-4974-a44c-3c7a699d8edf",
    "version": 3
}
```

可以看到，钥匙文件中只存储了以太坊账户私钥的密文（在 ciphertext 字段），只有知道用户自己设定的密码才能得到以太坊账户的真正私钥，该私钥用于此账户所有交易的签名。由于以太坊采用区块链这种去中心化技术，因此一旦钥匙文件丢失就意味着账户再也找不回来了。所以用户要确保已备份好自己的钥匙文件，并确保自己的密码不会外泄。

2. 创建账户

在第 2 章已经介绍过了以太坊的多种客户端。到目前为止，最普遍使用的是基于 Go 和 C++ 程序设计语言的客户端（即 go-ethereum 和 cpp-ethereum），因此在这一章我们主要采用 go-ethereum[5] 来演示如何操作以太坊账户。

通过 go-ethereum 创建账户十分容易。以 Linux 操作系统为例，创建一个账户需要以下几个步骤：

1）安装好以太坊和 go-ethereum，在客户端执行以下命令。

```
sudo apt-get install software-properties-common
sudo add-apt-repository -y ppa:ethereum/ethereum
sudo apt-get update
sudo apt-get install ethereum
```

2）执行 geth account new 命令创建账户，并设置密码。

```
geth account new
Your new account is locked with a password. Please give a password. Do not
    forget this password.
Passphrase:
Repeat passphrase:
Address: {850fa7796f372a5f6a7b59976ac5cca6e6565cbb}
```

设置好密码之后，客户端会显示所创建的新账户的地址，供以后使用。

3）之后执行 geth 命令，同步已有的所有区块。

```
geth
```

当区块链全部同步好之后，就可以进行挖矿并部署智能合约了。

除了以上命令行的方式之外，以太坊的开发者还开发了拥有图形界面的以太坊钱包

（Ethereum Wallet），以方便用户更容易地管理账户和部署智能合约。用以太坊钱包创建账户十分简单，首先从以太坊在 github 上的官方网页（https://github.com/ethereum/mist/releases）上下载相对应的操作系统的钱包程序压缩包，并解压程序，运行钱包。第一次运行钱包时会出现如图 7-1 所示界面。

图 7-1 以太坊钱包初始界面

此时，用户可以选择同步以太坊的主链还是其公共的测试链（testnet）。在测试链上，用户不需要长时间地挖矿就可以很快地获得大量测试用的以太币，并运行测试自己的智能合约。而在主链上，用户则需要花费大量计算资源挖矿才能获得标准的以太币。

在选择好同步的网络之后，用户可以选择是否同步完区块链后再进入主界面。其主界面如图 7-2 所示。

此时，用户可以通过界面上的提示，一步一步创建自己的账户，单击 ADD ACCOUNT 按钮。账户的创建和区块链的同步无关，因此用户可以一边同步，一边创建账户。

3. 账户的备份

备份以太坊的账户十分容易，只需要找到相应的以太坊目录即可。根据不同的操作系统，以太坊的文件存储的目录如下。

- Windows：C:\Users\username\%appdata%\Roaming\Ethereum
- Linux：~/.ethereum
- Mac：~/Library/Ethereum

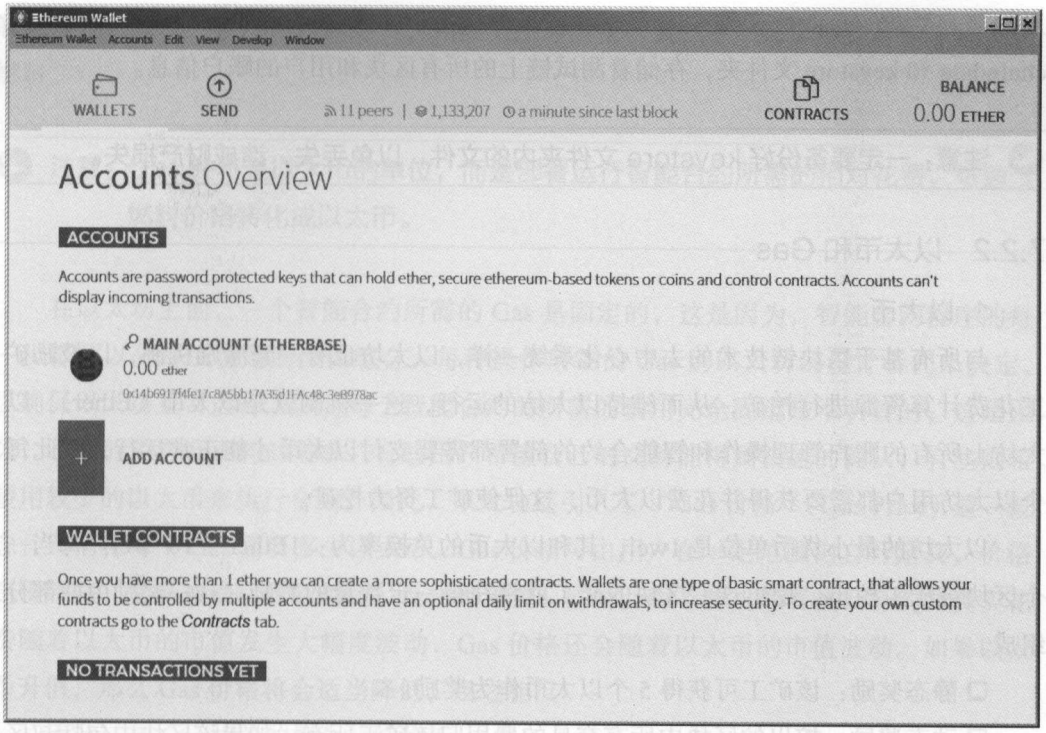


图 7-2 以太坊钱包主界面

以太坊文件的目录结构如下：

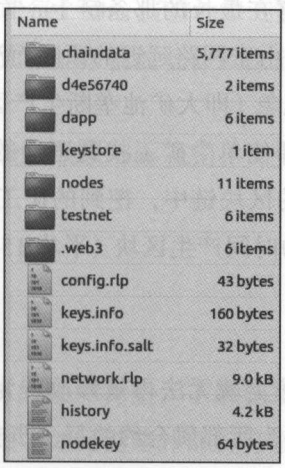


图 7-3 以太坊目录结构

其中，chaindata 文件夹存储着以太坊主链的所有区块，keystore 文件夹存储着用

户的账户数据，testnet 文件夹内有一套完整的用于以太坊测试链的文件，其中也含有 chaindata 和 keystore 文件夹，存储着测试链上的所有区块和用户的账户信息。

注意：一定要备份好 keystore 文件夹内的文件，以免丢失，造成财产损失。

7.2.2 以太币和 Gas

1. 以太币

与所有基于区块链技术的去中心化系统一样，以太坊也有一套激励机制，以鼓励矿工花费计算资源进行挖矿，从而维持以太坊的运行，这一机制就是以太币（Ether）。以太坊上所有的账户管理操作和智能合约的部署都需要支付以太币才能正常运行，因此每个以太坊用户都需要获得并花费以太币，这促使矿工努力挖矿。

以太坊的最小货币单位是 1wei，其和以太币的兑换率为：1Ether = 10^{18}wei。每当一个区块被矿工挖出，挖出这一区块的矿工就将获得一定数量的奖励。这一奖励由两部分组成。

- 静态奖励：该矿工可获得 5 个以太币作为奖励。
- 动态奖励：挖出的区块中所有交易的费用归该矿工所有；如果该区块中包括叔区块，那么矿工还可从每个叔区块中获得额外的 1/32 以太币作为奖励，但每个区块中最多只能包含 2 个叔区块。

这里，叔区块是指那些没有在最长的那条链上，而是在分叉链上所挖出的有效区块。挖掘这些区块的矿工可能是由于网路延迟的原因而没有同步到最新的区块。以太坊采用这种机制来分散中心挖矿现象（即大矿池垄断生产区块，导致单个的矿工总是落后于大矿池获得区块信息，因此即使单个矿工找到正确的区块，也无法获得任何收益）。一个叔区块一旦被包含在有效的区块链中，挖到的矿工可获得 4.375 以太币作为奖励。这也保证了以太坊能够以很短的时间产生区块（平均 15 秒），而不会因为网络同步的延迟而产生多个分叉。

2. Gas

智能合约一旦部署在以太坊上就无法再被修改。为了防止恶意用户部署无限循环运行的合约，以太坊要求用户要为所部属合约的每一步支付费用，而这些费用的基础单位就是 Gas。例如，部署智能合约，每一步需要支付 1Gas，停止合约不需要支付任何 Gas，创建合约需要支付 100Gas，而每次合约交易需要支付 500Gas[2]。因此，Gas 就相当于部署和执行智能合约所需要的燃料，没有燃料，就无法使用智能合约。这种燃料机

制维持着以太坊的经济体系的运行，用户只能通过挖矿或从矿工那里购买以太币来补充燃料。

> 注意：Gas 并不是以太币的单位，而是部署运行智能合约所需的相对花费，可通过燃料价格转化成以太币。

在以太坊上面，一个智能合约所需的 Gas 是固定的，这是因为，智能合约程序的每一步都可以分解成特定的操作组合，每种操作所需的 Gas 由以太坊的设计者们来决定，以确保以太坊正常运行。但将每个智能合约的所需的费用完全固定是不明智的，这是因为，不同的用户有不同的需求，有些用户希望自己的交易能得到快速的确认，有些则希望用较少的以太币来执行合约。因此，以太坊还引入了“Gas 价格”（Gas Price）这一概念，即消耗每个 Gas 需要多少以太币。Gas 价格可由用户在一定范围内自行定义，价格定得越高，交易被确认得就越快，反之则越慢。同时，为了防止部署执行合约的真实花费随着以太币的市值发生大幅度波动，Gas 价格还会随着以太币的市值波动。如果以太币升值，那么 Gas 价格将会适当降低，反之相反。

因此，与 Gas 相关的概念总结如下。

- Gas 花销（Gascost）：Gas 花销是静态的，其在针对某一种操作时是不变的。其目的是保证每种操作所需的计算资源保持不变。
- Gas 价格（Gasprice）：花费每个 Gas 所需的以太币的数量。Gas 价格可由用户自行调整，其基准价格随以太币的市值波动，以保证智能合约所需的真实花费不会出现大幅度变化。
- Gas 费用（Gasfee）：Gas 价格乘以 Gas 花销，即合约所需的真实费用，其单位是以太币。

7.2.3 合约和交易

1. 合约账户

我们在 7.2.1 节介绍了以太坊的账户类型，了解到现阶段以太坊的账户分为两类：外部所有账户和合约账户，并介绍了如何创建外部所有账户。这一节，我们重点介绍如何创建合约账户，换句话说就是如何在以太坊上部署和运行智能合约。以太坊的设计者们计划在以太坊发展的下一阶段取消这两类的账户的区别，将它们合并成一类账户。

以太坊的外部所有账户的主要具有以下几个特点：

- 可以存储以太币；
- 可以发起交易，其中包括交易以太币和部署运行智能合约；
- 用户创建账户密钥，并管理账户；
- 不支持智能合约代码。

与外部所有账户相比较，合约账户具有以下特点：

- 可以存储以太币；
- 可支持智能合约代码；
- 可响应别的用户或合约执行此智能合约的请求，并返回结果；
- 可调用别的智能合约。

在以太坊上，所有被记录在区块链内的活动都是由外部所有账户发起的。每当一个合约账户收到一个交易申请，其接收传递而来的参数，并通过运行在每个节点上的以太坊虚拟机（Ethereum Virtual Machine，EVM）执行自身的代码。每一笔有效的交易都将被记录在区块链上，通过所在区块在整个链的位置记录该交易的时间，整个区块链则反映了所有交易的执行顺序。

从形式上看，以太坊上的智能合约并不像传统合约那样需要得到合约方的履行，而看上去更像一种存在于以太坊网络中的“自治代理程序”（autonomous agents）。当这些程序接到申请，则总会按照已制定的程序代码来执行，并将其自身的状态变化永久地存储在区块链中。

在使用以太坊时，有两个概念需要区分。

- 交易（Transaction）：交易是指一个外部所有账户将一个经过签名的数据包发送到另一个账户的过程，这个过程中产生的账户状态变化将被存储到区块链上。
- 消息（message）：以太坊上的合约账户有能力向其他合约账户发送“消息”。这里的消息是一个虚拟的对象，并不会具体地存在以太坊的区块链内，可以将其想象成一个函数调用的过程。

本质上来说，交易和消息是两个非常相似的概念。区别在于，消息是由合约账户产生的，而交易是由外部所有账户产生的。因此，合约账户和外部所有账户一样，可以同其他合约账户产生联系。

2. 智能合约的编写

以太坊上的一个智能合约就是一段可被以太坊虚拟机执行的代码，这些代码以以太坊特有的二进制形式存储在区块链上，并由以太坊虚拟机解释，因此被称为以太坊虚拟机位码（bytecode）。

相较于其他可部署智能合约的区块链系统，以太坊的最大特色就是，以太坊虚拟机的建立使得智能合约的编写变得非常容易。这些智能合约通常可由高级语言编写，并通过虚拟机转化成位码存储在区块链上。目前来说，用于以太坊智能合约开发的语言主要有3种。Solidity、Serpent、LLL。

作为最流行的智能合约语言，Solidity以其简单易用和高可读性受到以太坊设计者们的推荐。目前，编译Solidity代码最简单的方式是使用在线的编译器（https://ethereum.github.io/browser-solidity/），也可以在命令行下使用solc编译器对代码进行编译。目前，很多编辑器和集成开发环境IDE（如Visual Studio）已开始支持Solidity代码的编写。此外，专门为以太坊设计的IDE也在不断开发完善中，例如，Ethereum Studio[3]和Mix IDE[4]。

3. 智能合约的部署流程

在部署合约时，以太坊虚拟机负责将用户编写的智能合约代码编译成位码。这些位码被存在区块链上，在需要时通过web3.js Javascript API调用，并可用来构建与之交互的Web应用。这些API由web3.js库提供，是和以太坊节点建立联系的媒介，其本质是通过JSON-RPC协议与本地的以太坊节点进行通信。这里，JSON是一个轻量级的、以文字为基础、易于阅读的数据存储和交换语言，其本质是JavaScript的一个子集，常作为Web应用的数据存储格式。JSON-RPC则是一个由JSON格式编码的、轻量级的远程过程调用协议（remote procedure call protocol），其定义了一些数据结构、规则、过程和接口，可用于网络上绝大多数的数据通信协议（如HTTP）。

总的来说，在以太坊上部署和运行智能合约需要以下几个步骤：

1）启动一个以太坊节点（如geth）。

2）使用智能合约语言编写智能合约（如Solidity）。

3）使用solc编译器将编写好的合约代码转换成以太坊虚拟机位码（如Browser-Based Compiler）。

4）将编译好的合约代码部署到网上需要消耗用以太币购买的GAS，并且需要合约发起用户使用自己的外部所有账户对将要部署的合约进行签名，通过矿工的确认后，将合约代码存于以太坊的区块链上。在这一步中，用户可获得合约的地址，以及调用合约所需的接口（interface），以便之后使用。

5）使用web3.js库所提供的JavaScript API接口来调用合约。这一步也会消耗以太币，具体消耗值取决于所调用的合约功能。

以太坊上的智能合约部署和调用的过程如图7-4所示。在7.4节中，我们将通过几

个简单的智能合约的实例，具体展示如何在以太坊上部署和运行智能合约。

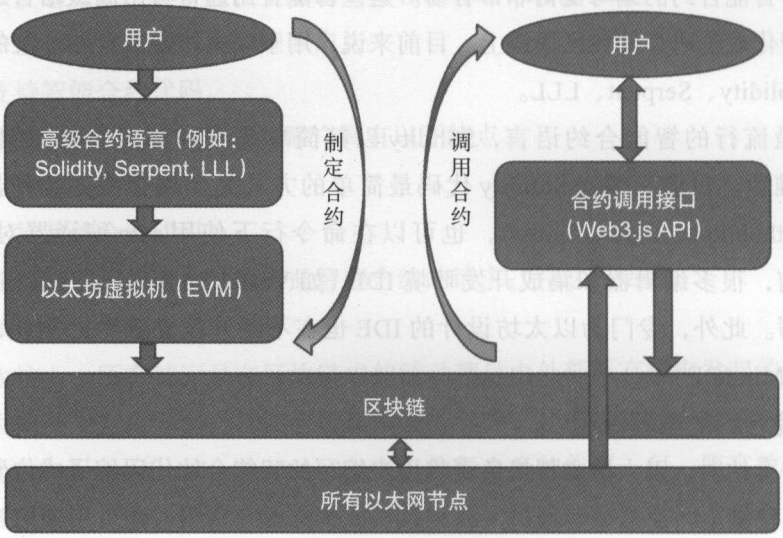


图 7-4 以太坊合约的部署和调用

7.3 以太坊虚拟机

以太坊并不是唯一一个可以在区块链上部署智能合约的平台（例如，很多智能合约都可以部署在比特币的区块链上），但使得以太坊与众不同的重要一点就是建立在区块链上的以太坊虚拟机。虚拟机的引入使得编写智能合约变得异常容易，高度脚本化的程序设计语言（如 Solidity）使得普通用户也能轻松地开发自己的智能合约，而不需要太多的专业学习。在未来，以太坊的设计者们还有更大的野心，他们试图建立一个类似于苹果电脑公司 App 商店的中心化 App（DApp）商店，这将极大地扩展以太坊的应用范围。

简单来说，以太坊虚拟机是建立在以太坊区块链上的一个代码运行环境，但虚拟机本身并没有存储在区块链内，而是和区块链一样同时存储于各个节点计算机上。每个参与以太坊网络中的校验节点都会运行虚拟机，并将其作为区块有效性校验协议的一部分。每个节点都会对合约的部署和调用进行相同的计算，并存储相同的数据，以确保将最权威（最真实）的结果记录在区块链内。

以太坊虚拟机是一个图灵完备的 256 位虚拟机，这说明以太坊虚拟机可以进行任何种类的计算。但为了防止恶意用户设计无限循环代码使虚拟机的运行瘫痪，以太坊虚拟机中执行的代码严格受到一个参数的制约，这个参数就是 Gas。这规定了可运行的计算

指令的数量上限，从而不会产生无限循环（无限循环最终会因耗尽 Gas 而中止）。

以太坊虚拟机的构架实际上是一个简单的堆栈式结构，每个堆栈项目为 256 位，即虚拟机的位宽为 256 位，其目的是使之能够方便地应用于 256 位的 Keccak 散列算法和椭圆曲线计算。堆栈的存储（storage）是一个基于字段地址的数组，其最大包含 1024 个元素。此外，虚拟机还包含一个独立的基于字段地址的内存，但不同于普通的内存模型，这个独立的内存是一个非易失性内存（non-volatile memory），即当虚拟机不运行时，其所存储的数据不会丢失。该内存中的记录作为整个以太坊系统状态记录的一部分。虚拟机的存储和内存在初始时都被设置为 0。

以太坊虚拟机还可以处理异常执行，其中包括堆栈溢出和无效指令等。同时，针对 GAS 不足的异常，虚拟机会立即停止工作，并将问题报告给交易处理器或运行环境的代理程序，由它们单独处理。

1. GAS 的消耗

在以太坊虚拟机内部，GAS 的消耗会出现在下列 3 种情况中（其中第一种情况最常见）。

1）当需要执行特定的内部抽象操作时，例如，运行 SHA3 散列运算时。

2）当进行一个从属的消息调用或合约创建时，例如，执行 CREATE、CALL 或者 CALLCODE 操作时。

3）当需要增加账户内存使用量时。

在账户进行操作时，需要支付费用的账户内存使用量应该是 32 个字节的整数倍，以保证使用的所有内存都能包括在计费范围内。例如，如果使用了 33 个字节的内存，那么账户需要支付两个 32 字节的费用。此外，内存使用计费机制还有助于激励用户使用较少的内存。当执行账户内存清理操作时，该操作不仅不会消耗任何 GAS，还会得到一定数量的内存使用费用的折扣，以鼓励用户尽量释放不用的内存。在实际操作中，这种折扣在账户执行之前就已经被支付给用户，这是由于内存初始化使用所产生的费用要高于一般的内存使用。

2. 虚拟机运行环境

假设整个以太坊网络的状态为 σ，合约运算剩余的 GAS 为 g，那么在整个运行环境中还有许多重要的信息。

- Ia：当前代码的合约地址；
- Io：发起这次合约交易的发起者地址；
- Ip：用户为这次交易设置的 Gas 价格；

- I_d：这次交易的输入数据，该输入的数据结构是一个数组；
- I_s：执行这次合约交易的账户地址；
- I_v：合约账户的余额；
- I_b：用于执行虚拟机代码所需的数组；
- I_H：目前区块的数据头；
- I_ε：目前执行的 CALL 操作和 CREATE 操作的数量。

假设以上信息都包含在一个元组 I 内，系统状态变化的函数是 Ξ，σ' 为系统运行后的状态，g' 为运行后剩余的 Gas，s 为执行终止（suicide）操作的合约列表，l 为记录序列，r 为运行后所返还的 Gas，o 为合约运行后所产生的输出，那么整个以太坊的状态转换可定义为以下公式：

$$(\sigma',\ g',\ s,\ l,\ r,\ o) = \Xi(\sigma',\ g,\ I)$$

3. 状态转换函数（Ξ）

为了完成整个以太坊系统的状态转化，需要定义状态的转换函数 Ξ。在大多数实际情况下，整个系统的状态转换是一个不断地迭代系统临时状态和虚拟机临时状态的过程。迭代的过程需要调用异常检查函数和指令输出函数。迭代的终止由以下两个条件决定：

- 系统状态是否出现异常而使虚拟机停止工作，其中包括 Gas 不足、指令无效、虚拟机堆栈容量不足等情况，任何正常的系统指令都不会造成异常状态的出现。
- 虚拟机在正常状态下停止工作，例如，所有指令执行完毕返回结果。

在每一次迭代过程中，智能合约的指令被压入堆栈，虚拟机按堆栈的索引执行指令。每执行一条指令，将支付相应的 Gas，直到所有指令执行完毕，堆栈被清空。其中如果遇到异常，虚拟机则停止工作逐层向上返回异常。

4. 区块链系统状态的验证

在以太坊虚拟机正确执行所有指令之后，系统的状态得以转换。为了保证这种转换权威而有效，每个以太坊节点都可能会对系统的状态进行验证，并达成共识，确认交易的有效性。在以太坊上，对于交易记录的信任建立在对最权威区块链的信任的基础之上。以太坊上最权威的区块链是在一个树结构中从根节点（root）到叶子节点（leaf）的路径。为了确认哪条路径是权威区块链，理论上来说，是找到哪条路径通过工作量证明花费的计算量最大，即最“重”的那条路径。实际情况当中，最权威的区块链是由从根节点（即起源区块）到某个叶子节点（即新生区块）间最长路径来决定的。这条路径越长，就意味着在这条路径上所消耗的计算资源越多，因而由于工作量证明，说明这条区

块链是最权威的，能够得到所有用户的认可。

每产生一个新的有效区块，以太坊系统需要以下几个步骤才能将该区块加入权威区块链上。

1）验证该新区块的 ommer 区块的有效性。这里 ommer 区块是指该新区块的“祖父”区块除当前新区块所在链的其他后继区块，即叔区块。每个区块中最多可包含两个 ommer 区块。

2）验证该新区块中所包含的交易的有效性，即所有交易所花费的 GAS 是否与该区块链中所标记的 GAS 花费量一致，并与每笔交易一一对应。

3）对相应的由于新有效区块的产生而能得到以太币奖励的账户发放奖励，其中包括挖到该区块的矿工账户和包含在该区块内的 ommer 区块所属的矿工账户。

4）验证该新区块链的工作量证明，并确认将新区块连接在权威区块链上，并将整个系统更新到最新状态。

7.4 实例：在以太坊上开发实施智能合约

前几节介绍了智能合约和在以太坊上部署运行智能合约的相关基础知识，这一节将通过实例展示如何在以太坊上部署运行一个真正的智能合约。通常来说，在以太坊上可以通过两种常用的方式部署运行智能合约：一种方式是使用图形界面的以太坊钱包，另一种方式是使用 go-ethereum 通过交互命令部署智能合约。下面将分别介绍这两种方法。

7.4.1 通过以太坊钱包部署智能合约

使用以太坊钱包部署智能合约并不需要太多的操作程序，除了需要使用合约语言编写智能合约之外，不需要编辑任何其他代码。其所有的合约部署和调用操作都可以在图形界面下完成，十分方便快捷。

1. 部署智能合约

以一个公司分配股份给权益人的智能合约 [6] 为例。通过以太坊钱包部署智能合约包括以下几个步骤：

1）下载最新的以太坊钱包或 Mist 浏览器 [7]。Mist 浏览器是未来实现发布浏览调用 DApp 的工具，现在还在开发之中，目前的版本集成了以太坊钱包，可在 Mist 浏览器内部使用钱包的所有功能。

2）运行以太坊钱包，按照之前在 7.2.1 节介绍的方法选择想要同步的区块链。为

了方便获得以太币进行测试，本节所有智能合约的相关操作都在测试网络（testnet）下进行。

3）按照 7.2.1 节介绍的方法创建用户外部所有账户，并等待测试区块链全部同步完成。

4）为了测试的方便，假设钱包内已有两个外部所有账户（即 MAIN ACCOUNT 和 ACCOUNT 1），每个账户中都有一些测试用的以太币。其当前状态如图 7-5 所示。

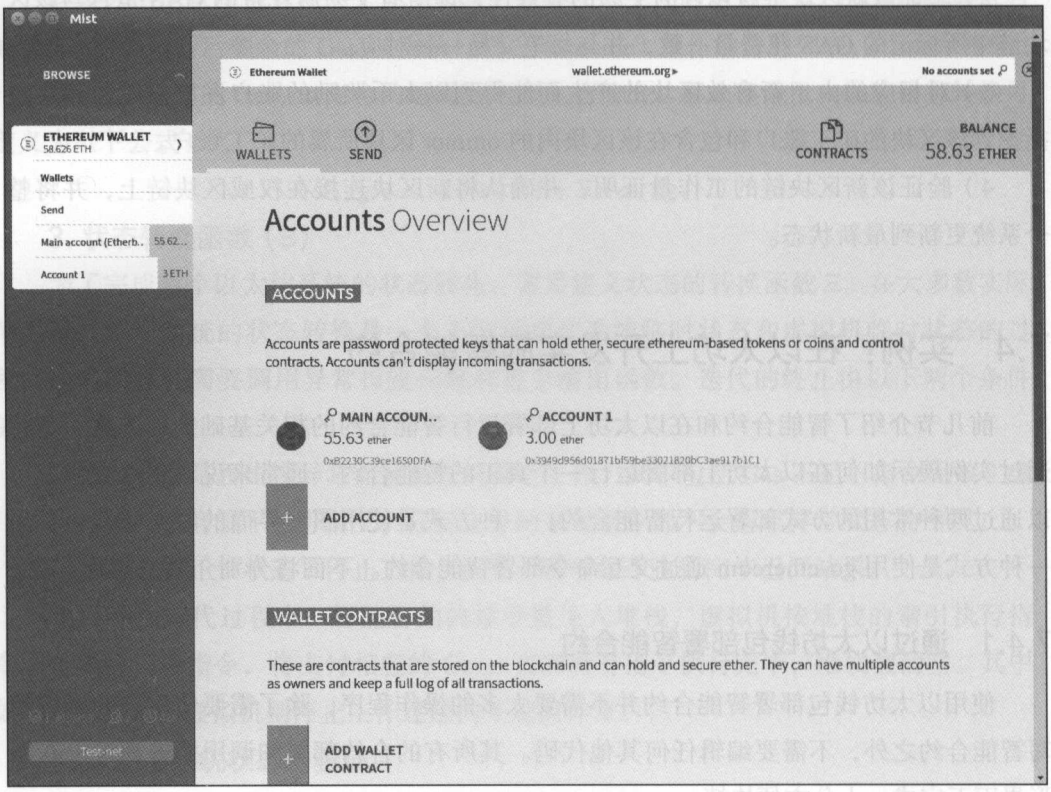


图 7-5　以太坊钱包当前状态

5）现在创建股权的合约。单击右上角的 CONTRACTS 按钮进入合约菜单，选择部署智能合约（Deploy contract），选择发起智能合约的账户（这里由 MAIN ACCOUNT 发起智能合约），并可在 AMOUNT 菜单中向将要创建的合约账户发送以太币。这里我们不需要合约账户支付以太币，所以将 AMOUNT 设置为 0。将用 Solidity 语言编写好的智能合约代码复制到 SOLIDITY CONTRACT SOURCE CODE 菜单。其合约代码如下：

```
contract MyToken {
```

```
    /* 在合约中使用 public 关键字定义所有能被别的合约访问的变量 */
    string public name;
    string public symbol;
    uint8 public decimals;

    /* 建立一个数组存储账户的余额 */
    mapping (address => uint256) public balanceOf;

    /* 建立一个公共的事件用于用户通知 */
    event Transfer(address indexed from, address indexed to, uint256 value);

    /* 初始化合约，当合约内的函数名和合约名相同时（MyToken），则该函数是合约的构造函数 */
    function MyToken(uint256 _supply, string _name, string _symbol, uint8
        _decimals) {
        /* 默认将股权分为 10000 份，即股权的最小单位是 0.01%*/
        if (_supply == 0) _supply = 1000000;

        /* 可自定义股权数量和最小单位 */
        balanceOf[msg.sender] = _supply;
        /* 定义股权名称 */
        name = _name;
        /* 设定股权所使用的单位符号，例如 %*/
        symbol = _symbol;

        /* 设定小数位数 */
        decimals = _decimals;
    }

    /* 创建股权转移函数 */
    function transfer(address _to, uint256 _value) {
        /* 检验是否有足够的股权 */
        if (balanceOf[msg.sender] < _value) throw;
        if (balanceOf[_to] + _value < balanceOf[_to]) throw;

        /* 更新股权转让信息 */
        balanceOf[msg.sender] -= _value;
        balanceOf[_to] += _value;

        /* 通知用户股权转让成功 */
        Transfer(msg.sender, _to, _value);
    }
}
```

6）在 SELECT CONTRACT TO DEPLOY 菜单中选择要部署的合约（即 My Token）。在 CONSTRUCTOR PARAMETERS 菜单中输入参数，其页面如图 7-6 所示。

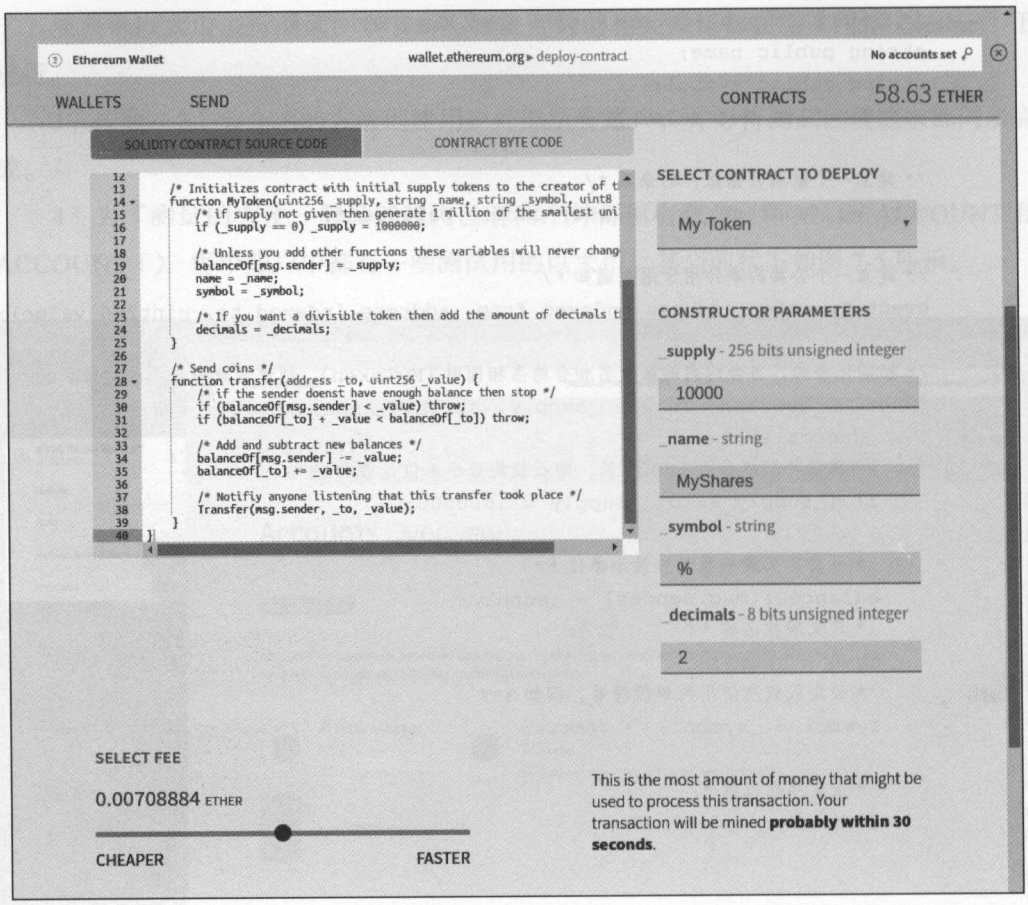


图 7-6 输入参数

7）从下拉菜单中可以看到部署合约的 Deploy 按钮和希望支付的交易费用，费用越多则该合约确认得越快，反之则越慢。单击 Deploy 按钮出现如图 7-6 所示的确认界面。

用户可在此界面中看到可能需要的交易费数量和当前 GAS 的价格，在 Data 菜单下可看见合约的以太坊虚拟机位码。之后单击 SEND TRANSACTION 部署合约。此时，在 WALLETS 界面下可看到刚刚发送出的合约正在接受确认。以太坊要求交易必须在 12 个区块产生之后才能得到最终确认，但只要有一个区块确认了合约，就可以调用合约的函数了（这不代表交易得到最终确认，只是临时确认）。其交易确认状态可从 WALLETS 界面下看到，如图 7-8 所示。

当 12 个区块确认完成时，合约才被真正保存到区块链中，即部署到以太坊的网络上，并可被调用。

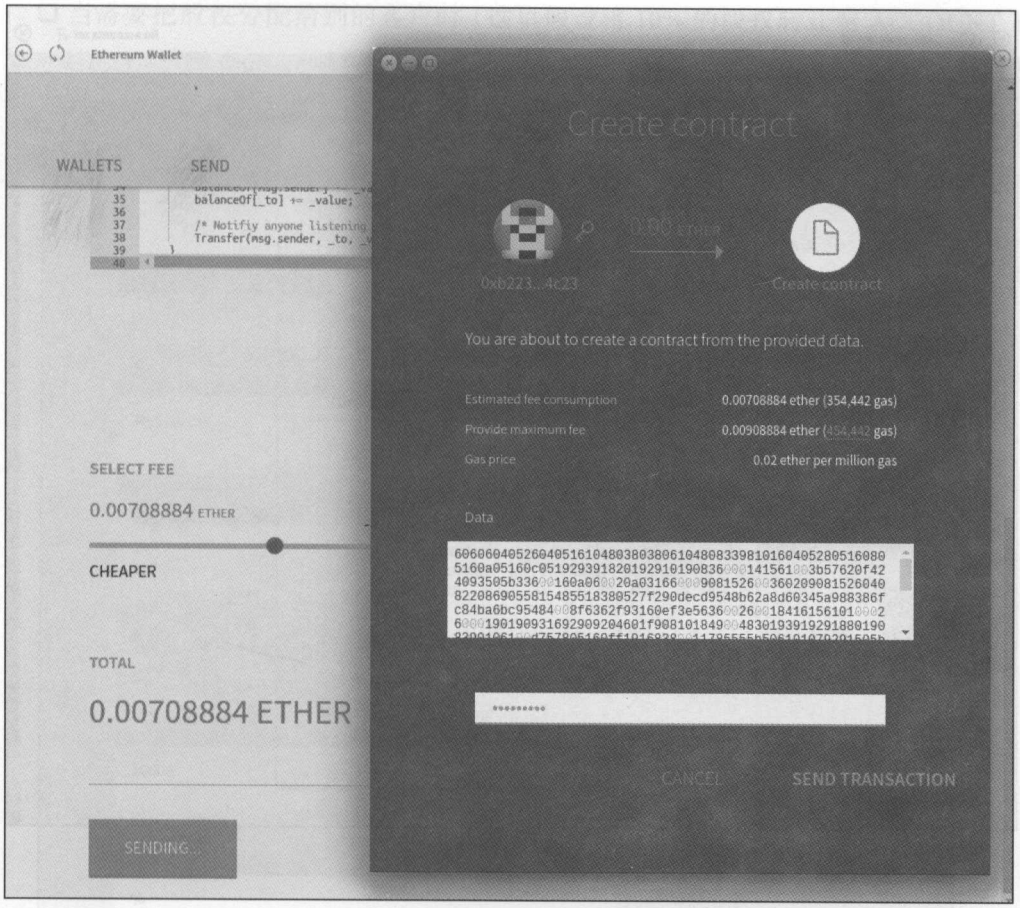


图 7-7　确认部署合约界面

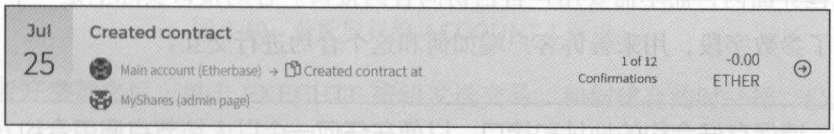


图 7-8　部署合约交易确认状态

2. 调用智能合约

在合约部署完之后，在钱包的 CONTRACT 菜单下可看到刚刚部署成功的合约（My Shares）。部署合约相当于创建了一个合约账户，因此在合约 My Shares 内可以看见返回的合约地址和合约接口（interface）。地址和接口是找到合约并调用合约的必要信息。如果用户希望在另外一个以太坊节点调用刚刚创建的合约，可在 Watch Contract 界面中输入合约的地址和接口，合约的名字可以任意选取，其状态如图 7-9 所示。

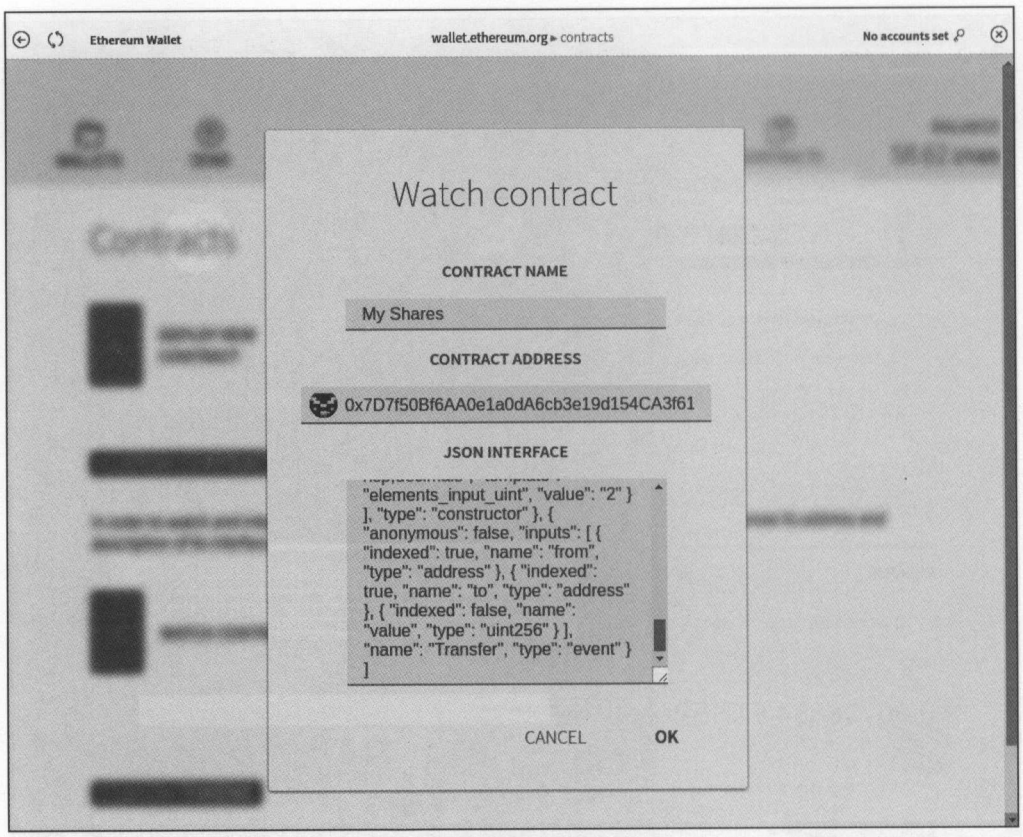


图 7-9 调用新合约

通过钱包调用合约也十分容易，这是因为以太坊钱包已经将合约接口的所有调用集成在钱包界面内，而不需要用户自己访问合约接口。合约接口实际上是一个 JSON 文件，定义了参数字段，用来告诉客户端如何和这个合约进行交互。

注意：请保存好合约的地址和接口，以便在任何一个以太坊节点调用合约。

单击 My Shares 合约进入合约界面，可看到所有该合约内部的公共参数和合约内定义的函数。这里我们只定义了 transfer 函数，它用于将股权分配给其他客户，还定义了一个名为 balanceOf 数组用来查询客户的股权。具体操作如下：

- 查询客户的股权余额只需将账户的地址输入 Balance Of 菜单下，就可看到该账户的股权余额。这里我们输入 MAIN ACCOUNT 账户的地址，就可看到数字 10000，这说明股权全部在 MAIN ACCOUNT 账户内，还未被分发出去。

- 当需要把股权分配给别的客户时（这里假设将 10% 的股权转让给 ACCOUNT 1 账户），选择 transfer 函数，并填入参数，其中包括接收账户的地址和要发送的股权数量，其状态如图 7-10 所示。

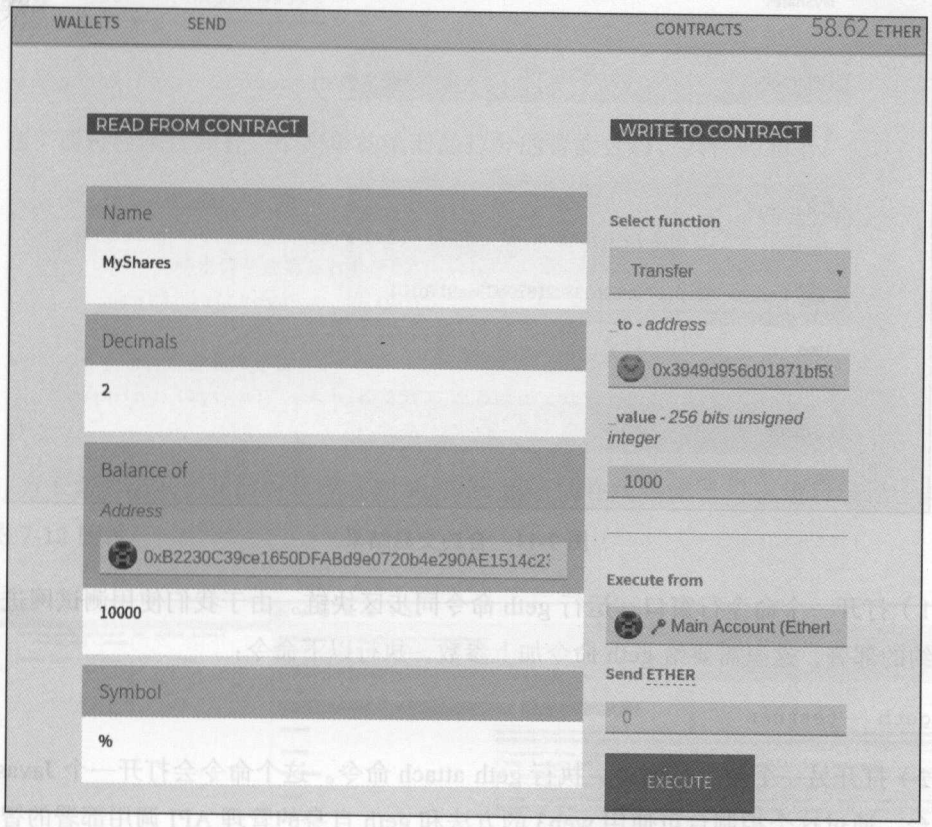


图 7-10　分配股权给 ACCOUNT 1 账户

在设置好参数之后，单击 EXECUTE 按钮发送交易。和创建合约时一样，经过 12 个区块的确认，该交易被写入区块链。此后在 My Shares 合约界面输入 ACCOUNT 1 账户的地址，可以看到数字 1000，说明 10% 的股份已转移给 ACCOUNT 1，如图 7-11 所示。

7.4.2　通过控制台部署智能合约

除了使用以太坊钱包外，用户还可以通过 web3.js Javascript API 在控制台命令行上部署调用智能合约。

1. 部署智能合约

在命令行部署智能合约需首先确认已经安装了 go-ethereum 客户端，其具体步骤如下。

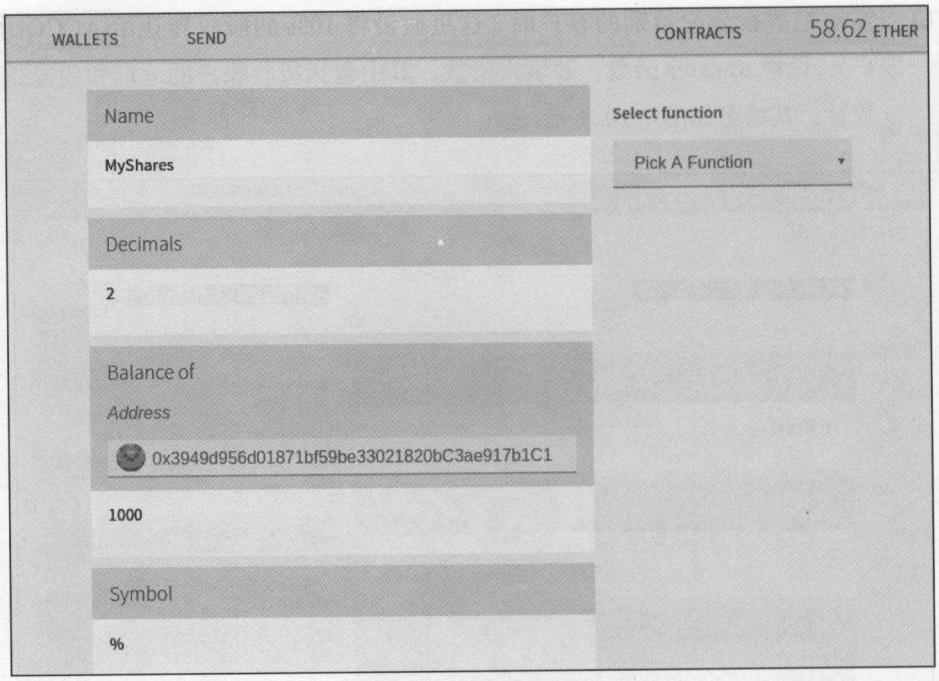


图 7-11 合约交易结果

1）打开一个命令行窗口，运行 geth 命令同步区块链。由于我们使用测试网进行智能合约的部署，这里需要给 geth 命令加上参数，执行以下命令：

```
geth --testnet
```

2）打开另一个命令行窗口，执行 geth attach 命令。这个命令会打开一个 Javascript 控制台，通过这个控制台可使用 web3 的方法和 geth 自身的管理 API 调用部署的智能合约。默认的 geth attach 命令打开的是附加于标准的以太坊区块链上的控制台。如需打开附加于测试链的控制台，则要明确为 geth attach 命令指明访问的位置，此时需要加入参数，执行以下命令：

```
geth attach ipc:/home/* 用户名 */.ethereum/testnet/geth.ipc
```

其中，geth.ipc 为以太坊的进程间通信接口，此接口用于测试网节点。同样在以太坊文件系统的主目录下也有一个 geth.ipc 文件，用于标准以太坊节点。

3）用户在部署合约之前，需要知道自己的账户地址和余额。在控制台输入以下命令可看到当前的所有外部所有账户：

```
personal.listAccounts
```

4）执行以下命令可以以以太币为单位查询账户的余额：

```
web3.fromWei(eth.getBalance("账户地址"), "ether")
```

此外，还需要解锁要发起智能合约的账户，解锁时需要输入账户创建时所设置的密码，其命令如下：

```
personal.unlockAccount("账户地址")
```

5）通过控制台部署一个简单的给商品打分的智能合约，其代码如下：

```
contract Rating {
    function setRating(bytes32 _key, uint256 _value) {
        /* 为特定编号的商品打分 */
        ratings[_key] = _value;
    }
    /* 显示特定商品的分数 */
    mapping (bytes32 => uint256) public ratings;
}
```

为了方便起见，我们将代码放到 Solidity 语言的在线编译器 [8] 上进行编译，其界面如图 7-12 所示。

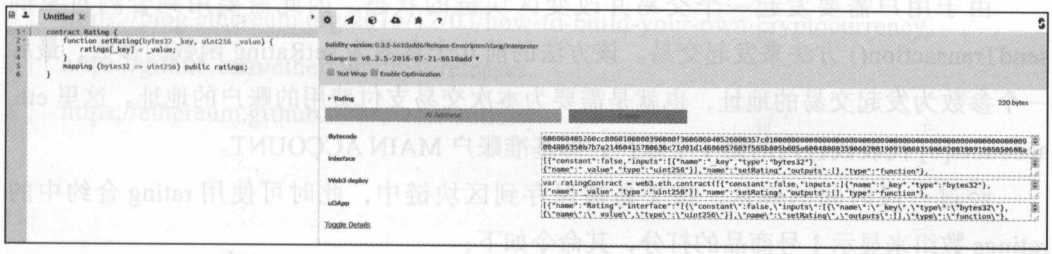


图 7-12 Solidity 在线编译器

编译完成之后，在右侧的菜单栏中将 Web3 deploy 菜单下的全部内容复制到控制台中，如图 7-13 所示。

经过一段时间的确认，控制台提示“Contract mined!”，则合约被创建成功，并返回合约的地址（address）和此次交易的散列值（transactionHash）。可通过这两个数值在区块链上寻找合约的信息。

2. 调用智能合约

在部署合约所使用的控制台窗口下，可直接使用合约名和函数名调用合约。假设一个用户想为 1 号商品打 3 分，需要调用 rating 合约的 setRating 函数，需要执行以下命令：

```
rating.setRating.sendTransaction(1, 3, {from: eth.accounts[0]})
```

图 7-13 在控制台部署智能合约

由于用户需要发起一个交易并改变区块链的状态，因此需要用到合约对象的 sendTransaction() 方法来发起交易。该方法的前几个参数为 setRating 函数的参数，最后一个参数为发起交易的地址，也就是需要为本次交易支付费用的账户的地址。这里 eth.accounts[0] 代表钱包内的第 1 个账户，即基准账户 MAIN ACCOUNT。

经过一段时间的确认，该交易被保存到区块链中，此时可使用 rating 合约中的 ratings 数组来显示 1 号商品的打分，其命令如下：

```
rating.ratings(1)
```

这时将会显示 3，表示 1 号商品被打了 3 分。由于此时并不需要改变区块链系统的状态，因此不需要使用 sendTransaction 方法，也就不需要支付任何费用。

如果想在其他以太坊节点通过控制台调用合约，则需知道合约的地址和接口。

执行以下命令实例化合约对象：

```
var NewRatingContract = eth.contract(interface).at(“address”)
```

这里，interface 的信息可从 Solidity 在线编辑器上获得，而 address 在合约部署之后返回得到。之后，使用 NewRatingContract.ratings(1) 命令就可查找到 1 号商品的打分。

7.5 本章小结

在这一章，我们首先介绍了什么是智能合约、智能合约的应用以及其起源。之后介绍了在以太坊上部署智能合约的基本知识以及背后的原理。接下来我们介绍了以太坊最大的特色——以太坊虚拟机的相关知识。最后，通过实例向读者分别展示如何用图形界面的以太坊钱包和控制台命令行部署运行智能合约。通过这章的学习，读者能够很快地了解智能合约，并顺利地在以太坊上部署自己的第一个智能合约。

参考资料

[1] Tapscott, Don; Tapscott, Alex (May 2016). The Blockchain Revolution: How the Technology Behind Bitcoin is Changing Money, Business, and the World. pp. 72, 83, 101, 127.

[2] Gavin Wood. Ethereum Yellow Paper: http://gavwood.com/paper.pdf.

[3] https://live.ether.camp.

[4] https://github.com/ethereum/mix.

[5] http://www.ethdocs.org/en/latest/.

[6] https://blog.ethereum.org/2015/12/03/how-to-build-your-own-cryptocurrency.

[7] https://github.com/ethereum/mist/releases.

[8] https://ethereum.github.io/browser-solidity.

第8章

超级账本项目

8.1 超级账本项目简介

8.1.1 项目背景

以比特币为代表的加密数字货币获得了巨大成功，活跃用户数量和交易量逐年增长。人们也渐渐意识到区块链技术的潜在价值，它不仅可以用作比特币的底层技术，还能够应用到更多的业务场景中。因而出现了很多利用比特币公有链的新型应用，如资产登记、公证等。但比特币的公有链无法克服自身固有的一些问题，例如，交易效率很低，整个网络吞吐量大约只有每秒7笔左右，而且每笔交易需要60分钟以上才能确认；另外就是交易的确定性（finality）问题也无法保证，从理论上讲，每个区块都是没有最终确定的。这些问题使得比特币的公有链不能满足大多数商业应用的要求。

为了克服上述不足，设计适合商用的区块链平台成为迫在眉睫的事情。在各界强烈的呼声中，Linux基金会于2015年12月启动了名为“超级账本”（Hyperledger）的开源项目，旨在推动各方协作，共同打造基于区块链的企业级分布式账本底层技术，用于构建支撑业务的行业应用和平台。超级账本将提供多种的区块链技术框架和代码，包含开放的协议和标准，不同的共识算法和存储模型，以及身份认证、访问控制和智能合约等服务。模块化、性能和可靠性是很重要的设计目标，用于支持各种各样的商业应用场景。

从创始成员看，参与超级账本项目的公司阵容相当强大，不仅有IBM、Intel、思

科等科技巨头，还有摩根大通、富国银行、荷兰银行等金融大鳄，还有 R3，ConsenSys 等专注区块链的公司。截至 2016 年 6 月底，超级账本项目已经汇集了全球超过 80 家公司，声势之浩大是其他技术联盟或开源项目无法比拟的。不管是从代码数量还是从社区参与度来看，超级账本都是最大的区块链开源项目。和比特币、以太坊等由极客主导的公有链项目相比，超级账本则是由大企业领衔的商业化联盟链项目。

8.1.2　项目管理形式

超级账本项目由会员公司组成。只要是 Linux 基金会的会员公司，缴纳一定的年费，即可成为超级账本项目的会员。按照所缴年费数额的多少，分为首要会员（Premier Member）和普通会员（General Member）。还有一种无需缴费但无投票权的附属会员（Associate Member）。成为超级账本项目会员后，可以参加日常会议，并享有会员特权和履行会员义务。

超级账本项目设有理事会（Governing Board），负责日常事务管理，包括审核预算、监督项目和市场活动、表决重要事项等职责。每个首要会员可以委派一名理事会成员，普通会员中每年可推选不超过 2 名理事会成员。技术指导委员会（Technical Steering Committee，TSC）主席以及一名用户顾问团（End User Technical Advisory Board，EUTAB）成员也是理事会成员。

技术指导委员会由项目的贡献者（Contributor）或维护者（Maintainer）选举产生，主要任务是在技术上保证项目正常进行，包括制定技术方向、审批项目议案、设立工作组和工作流程以及和技术社区用户交流等。任何人都可以成为项目的贡献者，只需贡献代码、文档或其他技术性产品。部分项目贡献者将成为项目的维护者，拥有对代码库的管理权。在超级账本项目开始的前 6 个月，技术指导委员会由每个首要会员指派的一名成员，以及各个顶级项目的维护者共同组成。TSC 还会选举出一位主席，作为理事会成员，以加强 TSC 与理事会的沟通。

超级账本项目还设立了市场委员会（Marketing Committee）和用户顾问团。其中，用户顾问团有一名代表可以参与理事会的讨论和投票。

超级账本项目的目标是为商业区块链应用提供底层支持，因此，在知识产权上采用了商业友好的使用许可。所有添加到项目中的代码都要使用 Apache V2.0 的许可协议，项目对外提供的代码同样依照 Apache V2.0 的许可协议，这是非常宽泛的许可协议，可以满足绝大部分商业应用的需求。项目的文档遵循知识共享 4.0 国际许可协议（Creative Commons Attribution 4.0 International License），适合商业和非商业用途。

8.1.3 项目的生命周期管理

超级账本里包括很多不同的项目（project），每个项目是社区在某方面协同努力的工作内容，既可以是创建各类文档，也可以是开发特定功能的代码。超级账本采用了开源项目常见的孵化流程：一方面鼓励社区提出更多的新建议；另一方面给社区提供项目进展情况的指引，以便了解项目是否已经成熟可用，或处于试验或开发阶段。

超级账本项目根据发展程度可处于 5 种状态，分别是：提案、孵化、成熟、弃用和终止。项目在开展的过程中，可能会在数个状态之间转换多次。

（1）提案

提案（Proposal）就是设立项目的建议，任何人都可以向技术指导委员会递交提案。提案需要有清晰的描述和项目的范围，确认将投入开发的资源和项目维护者，同时必须是厂商中立的方案。如果 TSC 批准了提案，该项目就正式启动，交由相关的项目维护者管理，项目也就进入了孵化状态。

（2）孵化

进入孵化（Incubation）状态的项目，可在超级账本的 Github 账号下创建专属的代码库，以便社区能协作开发、共同探索不同的方案，为项目添加所需的各种功能。超级账本同时包含多个孵化期的项目，为了鼓励社区的创新，项目之间或许有重叠的部分。长远看，最终可取长补短，把项目间共性或互补的功能抽取合并到同一个项目的框架中，实现完整的技术方案。

孵化项目的目标就是使代码达到质量稳定、可用的标准，具有成熟的发布流程，并在社区拥有众多的活跃开发者。项目的维护者可向技术指导委员会提出审批申请，宣布项目转变为成熟的状态。当然，项目由于实施不当或目标改变等原因，也有可能最后无法从孵化状态转化为成熟状态。

（3）成熟

从孵化状态“毕业”的项目将进入成熟（Mature）状态，项目的成果适合在实际的应用中使用。和大多数开源项目一样，成熟状态的项目还会持续地完善功能、修复错误，以及定期发布更新版本。

（4）弃用

项目发展到一定阶段，由于各种原因，已经不适应实际需要，此时项目维护者可投票表决，是否让项目进入弃用（Deprecated）状态。投票如果通过了弃用决定，技术指导委员会将宣布项目进入弃用状态。社区将继续维护该项目 6 个月，之后将不再发布任何更新。

（5）终止

在弃用状态持续 6 个月后，项目正式进入终止状态（End of Life），不再维护和开发。

8.1.4　项目发展状况

超级账本的初始成员公司中，不少已经开发了自己的区块链项目，他们都希望贡献这些代码给超级账本，成为其中的项目。这些成员公司的备选项目功能上既有侧重，也有重复，因此，较好的方式是把这些项目整合，互通有无，形成功能完整统一的方案。

截至 2016 年 7 月，通过提案进入孵化状态的项目有两个：Fabric 和 Sawtooth Lake（锯齿湖）。Fabric 是由 IBM、数字资产和 Blockstream 三家公司的代码整合而成。由于这三家公司原来的代码分别使用不同的语言开发，因此无法直接合并到一起。为此，三家公司的程序员进行了一次黑客松编程[⊖]。通过这次黑客松编程，终于把原来用不同语言编写的 3 个项目集成到一起，可实现基本的区块链交易和侦听余额变化的功能。这次黑客松的成果奠定了 Fabric 项目的基础。Sawtooth Lake 来自 Intel 贡献的代码，是构建、部署和运行分布式账本的高度模块化平台。该项目主要提供了可扩展的分布式账本交易平台，以及两种共识算法，分别是时间消逝证明（Proof of Elapsed Time，PoET）和法定人数投票（Quorum Voting）。

随着更多的提案通过审批，超级账本会包含越来越多的项目。本章主要介绍已经进入孵化状态的两个项目：Fabric 和 Sawtooth Lake。

8.2　Fabric 项目

8.2.1　项目概述

Fabric（编织品）项目的目标是实现一个通用的权限区块链（Permissioned Chain）的底层基础框架。为了适用于不同的场合，采用模块化架构，提供可切换和可扩展的组件，包括共识算法、加密安全、数字资产、记录仓库、智能合约和身份鉴权等服务。Fabric 克服了比特币等公有链项目的缺陷，如吞吐量低、无隐私性、无最终确定性以及共识算法低效等，使得用户能够方便地开发商业应用。

在超级账本联盟成立之前，IBM 公司就已经开源了一个叫作“开放区块链”（Open Blockchain，OBC）项目。在联盟成立之后，IBM 把 OBC 项目约 44 000 行代码贡献给

⊖ 黑客松是“黑客马拉松”的简称，它指程序员们集中到一起，花数天时间开发某些应用的编程活动，很多科技公司用这种方式激发员工的创新。

了 Linux 基金会，这部分代码成为了 Fabric 的代码的主要组成部分。在 2016 年 3 月的一次黑客松编程活动中，Blockstream 和数字资产两个成员公司把各自的区块链功能代码融合到 OBC 中，最终建立了 Fabric 的雏形，也就是 Fabric 项目进入孵化阶段的基础代码。

8.2.2 应用场景

超级账本有个重要的设计原则是按照“用例驱动”(use case driven) 的方式来实现的，所有功能都应该有对应的用例需求。鉴于超级账本是个通用型框架，无法预先确定将来所有的应用场景，因此，定义出部分典型的用例，可使超级账本先满足这部分代表性的区块链应用需求，然后再用可替换模块来满足其他需求。目前，Fabric 项目主要针对下面几种用例：金融资产管存、公司行为、供应链、主数据管理以及分享经济。需要指出的是，这些用例并非一成不变，随着项目的推进，可能会有所调整和增减。

（1）金融资产管存

金融行业最关心的区块链应用估计是资产的分布式管存，因为把资产（如证券）数据存放在区块链网络后，资产的利益相关人可以直接访问资产数据，而无需经过传统的中间人，可大幅度提高效率和节约成本。资产的交易可准实时地完成，交易的人员也能近乎实时地查询到相关资产信息。资产利益人可赋予资产自动执行的业务规则，从而进一步降低运营成本。与公有区块链应用的较大区别是，金融资产及其相关的交易、业务规则通常是保密的，例如，资产的余额只有持有人才能知道，其他人无法查看。

（2）公司行为

公司行为通常是上市公司发起的有关公司证券的事件，一般和股东有关，有时需要股东做适时的回应，例如要约收购、分红扩股、收购合并等。不管有多少中间机构在处理的流程当中，公司需要及时地把事件的完整信息递交给所有股东。当股东作出某项决定后，该结果会实时处理或清算（如发行新股等）。在整个事件处理过程中，应该保护股东的隐私，以确保投资者所作决定不受外界因素的左右。

（3）供应链

在供应链中，所有的参与者都通过区块链记录、追踪和共享各种数据，例如原材料来源、零部件检测结果以及货物的出处等。这些数据记录在区块链里，并贯穿于货物的生产、运输和销售等环节，从而提供深度回溯查询等核心功能。

（4）主数据管理

在很多的行业里，不同的组织之间往往共享一些主数据（Master Data）。例如，不

同的移动运营商之间，需要共同维护一份发射机站地理位置的数据。虽然主数据不是交易类型的商业信息，但是作为各组织间唯一的全局性数据，采用分布式的区块链来保证数据的质量和完整性具有非常重要的意义。

（5）分享经济

分享经济是指将闲置或没有被充分利用的实物资源分享出来，有偿地供陌生人暂时使用的一种商业模式。这种崭新的模式已经延伸到日常生活的多个领域，包括交通、住宿、保健、零售、教育等行业。在分享经济的模式下，最需要解决的就是陌生人之间的信任问题，即资源的提供方和资源的租用者，如何在缺乏信任的基础上安全地完成交易。目前主要的手段是通过分享经济平台来确保信任度。分布式区块链将是全新的一种去信任的方式，它不使用任何中间平台，便可达到各方参与者可靠交易的目的。

8.2.3　项目架构

Fabric 的逻辑架构如图 8-1 所示，底层由 4 种服务构成：身份服务、策略服务、区块链服务和智能合约服务。在这些服务的基础上为上层应用提供编程接口（API）、软件开发工具（SDK）以及命令行工具（CLI）。

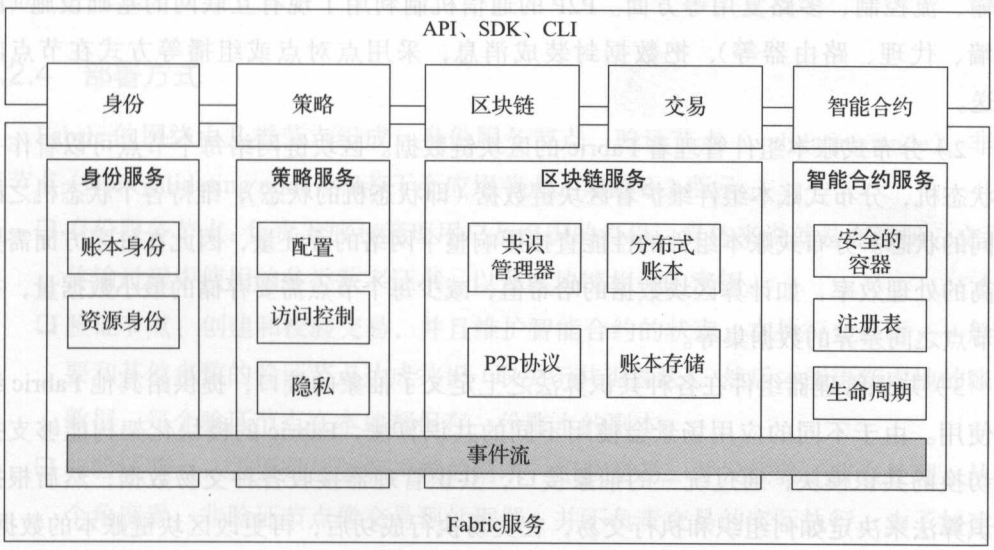


图 8-1　Fabric 项目的架构

1. 身份服务

Fabric 是权限区块链（Permissioned Chain），与比特币、以太坊这类匿名的无权限区

块链网络最大区别就是具有身份识别能力。在 Fabric 账本各类事件和交易中，参与者和对象都具有明确的身份信息。身份服务（Identity Service）管理着系统中各种实体、参与者和对象的身份信息，包括参与的组织、验证者和交易者，账本中的资产和智能合约，系统组件（网络、服务器）以及运行环境等。验证者在 Fabric 网络建立的时候可以确定参加交易的权限级别。

2. 策略服务

Fabric 里面许多功能需要用策略（policy）方式驱动，因此有独立的策略服务来提供系统的策略配置和管理功能。策略服务最重要的是访问控制和授权功能，Fabric 的交易通常要求参与方具有相关权限才能进行。其他的策略还包括加入和退出网络的策略，身份的注册、验证、隐私和保密的策略，共识策略等。

3. 区块链服务

Fabric 的区块链服务提供构建分布式账本最基础的能力，实现数据传输、共识达成等底层功能，并且提供发布 / 订阅的事件管理框架，分布式账本内部的各种事件可通知到外部监听的应用。Fabric 的区块链服务主要包含 4 个组件：P2P 协议组件、分布式账本组件、共识管理器组件和账本存储组件。

1）P2P 协议组件主要提供区块链节点之间直接双向通信的能力，包括流式数据传输、流控制、多路复用等方面。P2P 的通信机制利用了现有互联网的基础设施（防火墙、代理、路由器等），把数据封装成消息，采用点对点或组播等方式在节点间传送。

2）分布式账本组件管理着 Fabric 的区块链数据。区块链网络每个节点可以看作一个状态机，分布式账本组件维护着区块链数据（即状态机的状态），维持各个状态机之间相同的状态。分布式账本组件的性能直接影响整个网络的吞吐量，因此在许多方面需要较高的处理效率，如计算区块数据的哈希值，减少每个节点需要存储的最小数据量，补足节点之间差异的数据集等。

3）共识管理器组件在各种共识算法之上定义了抽象的接口，提供给其他 Fabric 组件使用。由于不同的应用场景会使用不同的共识算法，Fabric 的模块化架构能够支持可切换的共识模块，通过统一的抽象接口，共识管理器接收各种交易数据，然后根据共识算法来决定如何组织和执行交易，在交易执行成功后，再更改区块链账本的数据。Fabric 提供了 PBFT 共识算法的参考实现。

4）在区块链上保存大文件等数据是非常低效的操作，因此，通常大文档要存放在链外存储中。账本存储组件提供了链外数据的持久化能力，每个链外文档的哈希值可保

存在链上，从而保证链外数据的完整性。

4. 智能合约服务

Fabric 的智能合约（smart contract）曾经称为链上代码（chaincode），其实质是在验证节点（validating node）上运行的分布式交易程序，用以自动执行特定的业务规则，最终会更新账本的状态。智能合约分为公开、保密和访问控制几种类型。公开合约可供任何一个成员调用，保密合约只能由验证成员（validating member）发起，访问控制型合约允许某些批准过的成员调用。智能合约服务为合约代码提供安全的运行环境以及合约的生命周期管理。在具体实现中，可以采用虚拟机或容器等技术，构造安全隔离的运行时。

5. 应用编程接口

Fabric 项目的目标是提供构建分布式账本的基本能力，如账本数据结构、智能合约执行环境、模块化框架，网络通信等。用户可以在 Fabric 基础之上调用应用编程接口（API），实现丰富的应用逻辑，灵活易用的 API 将大大促进围绕 Fabric 的生态系统的发展。Fabric 的主接口采用 REST API，基本与 Fabric 服务相对应，API 分为身份、策略、区块链、交易（对应区块链服务）和智能合约等几类。为了方便应用开发，Fabric 还提供命令行接口（CLI），可覆盖部分 API 的功能，方便测试智能合约代码以及查询交易状态。

8.2.4　部署方式

Fabric 的网络由几类节点组成：身份服务节点、验证节点（validating node）、非验证节点（Non-validating node）和若干个应用节点，如图 8-2 所示。

- 身份服务节点：负责发放和管理用户及组织的身份，具体来说就是在注册、交易、传输过程中使用的各类数字证书，以及区块链相关的密钥。
- 验证节点：创建和校验交易，并且维护智能合约的状态。在执行交易时，一般需要和其他多数的验证节点达成共识（取决于共识算法），然后才能更新本地的账本数据。每个验证节点在本地都保存一份账本的副本。
- 非验证节点：主要是接收客户端的请求，组装交易，并发往验证节点处理，从这个角度看，非验证节点像交易预处理器，并不负责交易的实际执行。为了加速客户端的查询响应速度，非验证节点在本地也保留一份账本数据的拷贝。
- 应用节点：主要提供用户端（例如浏览器或移动设备）的后台服务，在收到请求后，把交易请求直接发往（或经由非验证节点转发）验证节点处理。

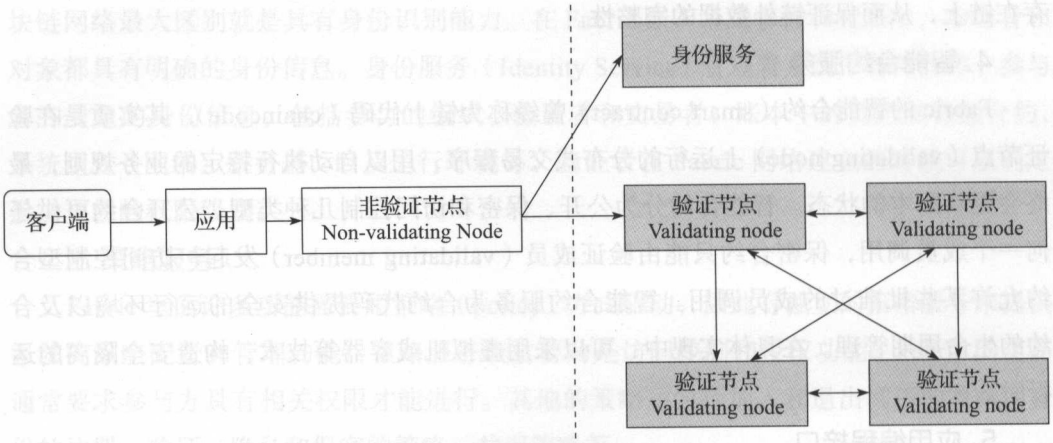


图 8-2 Fabric 的网络节点及拓扑

Fabric 的部署方式按照实际需要可有多种形式。由于 Fabric 是联盟链，组成网络的节点分别属于不同的联盟成员，只要这些节点可通过网络互相连接，每个成员能够选择自己节点的部署方式：既可把节点部署在自有的数据中心，也可把节点部署到公有云中。如果在云端部署节点，需要更强的加密手段来防止公网潜在的恶意攻击。由于 Fabric 节点部署的多样性，规划的时候应该把通信延迟、网络故障、节点失效、网络恢复等因素综合考虑在内，以符合应用的要求。

8.2.5 交易的执行

Fabric 上的交易（transaction）分成两种：部署智能合约和执行智能合约，智能合约可以看作部署在账本上的应用代码。Fabric 客户端可以通过 API 提交应用代码给任意一个验证节点，如图 8-3（a）所示。该验证节点在确认是有效的应用代码后，将该应用同步到其他验证节点中。通过这种分发机制，应用的代码最终会在各个验证节点保存一份，如图 8-3（b）所示。

应用的代码执行示意图如图 8-4 所示。步骤如下：

1）客户端发送执行请求给任意一个验证节点；

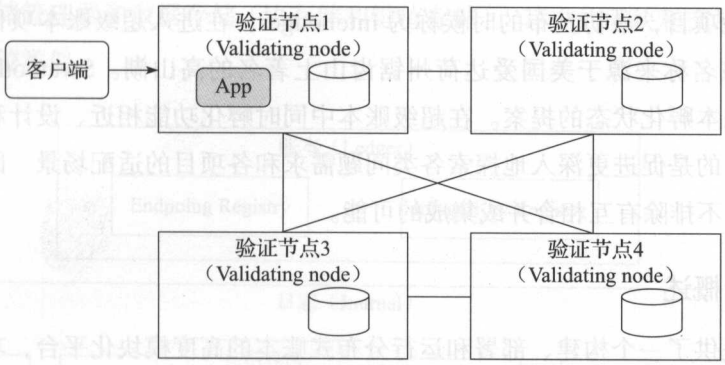
2）验证节点收到请求后，向本地账本（ledger）发送启动交易的指令；

3）验证节点创建隔离的运行环境，启动应用（智能合约）的代码；

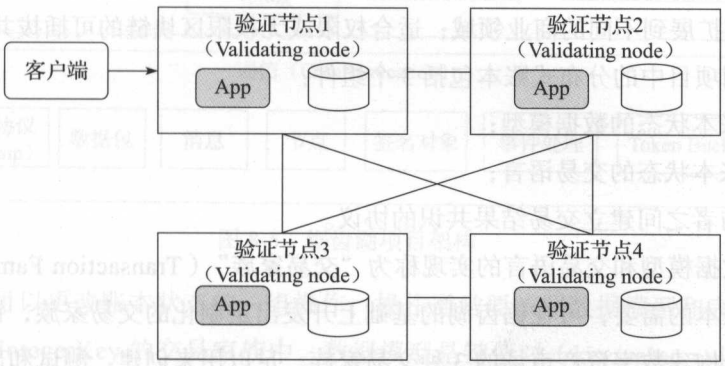
4）应用执行过程中，更新本地账本的状态；

5）应用完成后，验证节点向本地账本确认交易；

6）验证节点向其他验证节点广播交易。



a）应用提交到一个验证节点



b）应用同步到其他节点

图 8-3　应用代码的发布过程

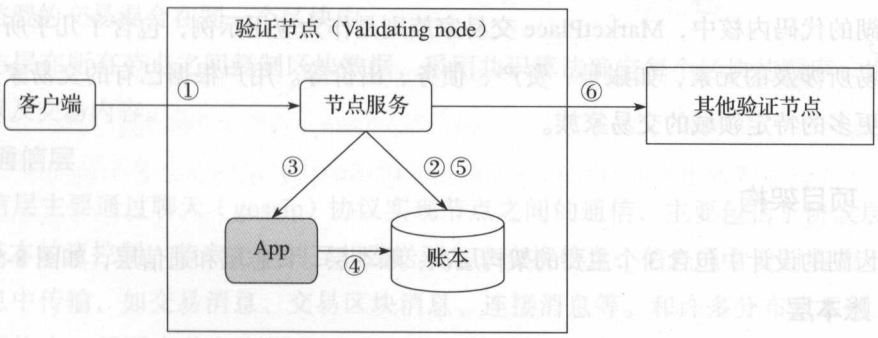


图 8-4　应用代码执行步骤

8.3　Sawtooth Lake 项目

Sawtooth Lake（锯齿湖）是由超级账本联盟成员英特尔（Intel）公司发起的分布式

账本平台试验项目，最初发布的时候称为 intelledger，在进入超级账本项目后，更名为“锯齿湖”，该名称来源于美国爱达荷州锯齿山上著名的高山湖。Swatooth Lake 是第 2 个进入超级账本孵化状态的提案。在超级账本中同时孵化功能相近、设计和实现不同的多个项目，目的是促进更深入地探索各类问题需求和各项目的适配场景。因此，在这些项目的后期，不排除有互相合并或集成的可能。

8.3.1 项目概述

锯齿湖提供了一个构建、部署和运行分布式账本的高度模块化平台，功能上有其独特的地方。如，锯齿湖分离了账本和交易，使两者成为松耦合的关系；提出了交易家族的概念，能够扩展到不同的商业领域；适合权限或无权限区块链的可插拔共识算法。

在锯齿湖项目中的分布式账本包括 3 个组件：

- 代表账本状态的数据模型；
- 改变账本状态的交易语言；
- 在参与者之间建立交易结果共识的协议。

其中，数据模型和交易语言的实现称为“交易家族”（Transaction Family）。尽管用户根据自身账本的需要，可在锯齿湖的基础上开发出定制化的交易家族，锯齿湖项目还是提供了适合构建数字资产市场的 3 种交易家族，足以用来创建、测试和部署这类市场应用。这 3 种可直接使用的交易家族分别为：注册账本服务（EndPointRegistry）、测试部署账本（IntegerKey）和数字资产买卖交易系统（MarketPlace）。前两种交易家族内置在锯齿湖的代码内核中，MarketPlace 交易家族则是作为应用示例，包含了几乎所有数字资产交易所涉及的元素，如账号、资产、债务、出价等。用户根据已有的交易家族，能够开发更多的特定领域的交易家族。

8.3.2 项目架构

锯齿湖的设计中包含 3 个主要的架构层次：账本层、日志层和通信层，如图 8-5 所示。

1. 账本层

账本层从概念上讲是交易类型的数据模型层次。因为其实现基本上通过延展日志层和通信层的功能来完成。例如系统内置的 Endpoint Registry 和 IntergerKey Registry 两个交易家族，以及作为范例的 MarketPlace 交易家族，都是通过扩展底层功能而来的。

2. 日志层

日志层是锯齿湖实现区块链核心功能的层次，实现了共识算法、交易（transaction）、

区块、全局存储管理器和数据存储（块存储和键值存储）。其中的区块和交易概念与其他区块链项目比较类似。

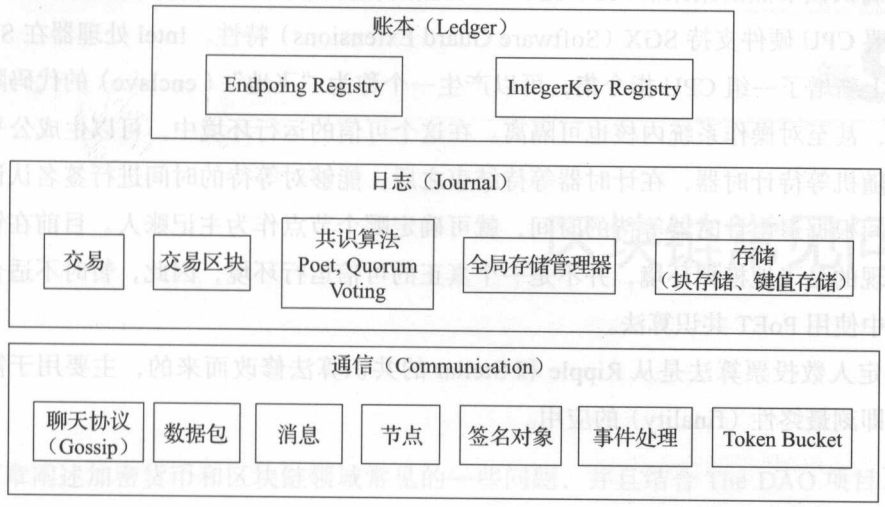


图 8-5　锯齿湖项目架构

交易是指可以更改账本状态的一组操作，操作通常要依照数据模型和表示形式的定义。例如，在 IntegerKey 的交易家族中，数据模型是键值对（key/value pair）存储，交易的操作可用设置（set）、增加（inc）和减少（dec）来表达。

区块则是一组交易的集合，是写入账本的单位。在锯齿湖日志的每个区块中，允许有不同类型的交易混合在同一个区块中。

日志层在所有节点之间复制区块数据，采用共识算法确定每个区块的顺序、块内交易顺序以及交易内容。

3. 通信层

通信层主要通过聊天（gossip）协议实现节点之间的通信，主要包括了协议层连接管理和基本的流控制。节点之间的互相发送消息来交换信息，信息通常要封装在不同类型的消息中传输，如交易消息、交易区块消息、连接消息等。和许多分布式系统一样，在整个架构中，需要在节点之间通过聊天协议发送大量的消息，为此，通信层实现了 Token Bucket 的机制，以控制数据包的传输速度。

4. 共识算法

锯齿湖项目使用的共识算法有两种：时间消逝证明（Proof of Elapsed Time，PoET）和法定人数投票（Quorum Voting）。

PoET 和比特币的工作量证明（Proof of Work，PoW）一样都属于彩票算法，即按照一定规则随机地选取出“赢家”节点，由该节点作为区块的主记账者，其他节点则负责验证和确认该节点的结果。PoET 比 PoW 明显的优势是不需要消耗大量的算力和能耗，但是需要 CPU 硬件支持 SGX（Software Guard Extensions）特性。Intel 处理器在 Skylake 微架构上新增了一组 CPU 指令集，可以产生一个称为“飞地”（enclave）的代码隔离运行环境，甚至对操作系统内核也可隔离。在这个可信的运行环境中，可以生成公平且可验证的随机等待计时器，在计时器等待结束之后，能够对等待的时间进行签名认证。各个节点间根据退出计时器等待的时间，就可确定哪个节点作为主记账人。目前在锯齿湖里面实现的是飞地模拟环境，并不是一个真正的可信运行环境，因此，暂时不适合在生产环境中使用 PoET 共识算法。

法定人数投票算法是从 Ripple 和 Stellar 的共识算法修改而来的，主要用于需要满足交易即刻最终性（finality）的应用。

8.4 本章小结

超级账本是目前最大的区块链开源项目，集结了众多科技和金融界的巨头，目标是建立面向商业应用的分布式账本基础技术。本章介绍了超级账本项目的产生背景和管理方式，并详细介绍了两个孵化期的项目 Fabric 和 Sawtooth Lake 的架构原理。Fabric 和 Sawtooth Lake 都提供了分布式账本的实现，两者都采用了可扩展和可插拔的模块化设计，以适应不同场景的需求。Fabric 侧重于权限控制、私密性保护和交易性能提高，Sawtooth Lake 则注重于提供完整的交易家族和节能的共识算法。超级账本成立时间较短，孵化期的项目发展过程中可能会有较大的变化，同时新的提案和项目也会不断增加。本章主要描述超级账本项目设计的总体原理和技术要点，旨在起到抛砖引玉的作用。

参考资料

[1] 超级账本项目：https://www.hyperledger.org/.

[2] 超级账本孵化项目 Fabric：https://github.com/hyperledger/fabric.

[3] 超级账本孵化项目 Sawtooth Lake：https://github.com/hyperledger/sawtooth-core.

[4] PBFT 共识算法，Miguel Castro and Barbara Liskov：http://dl.acm.org/citation.cfm?id=296824.

[5] Ripple 共识算法：https://ripple.com/files/ripple_consensus_whitepaper.pdf.

第9章

区块链常见问题

本章阐述加密货币和区块链领域常见的一些问题，并且结合 The DAO 项目，分析部分问题的产生原因和应对方法。其中，加密货币部分较多地使用比特币作为典型的例子来说明相关的原理。

9.1 钱包的安全性问题

像比特币这样加密货币（Cryptocurrency），因为采用无政府的去中心化方式发行，大多数国家都不承认它是合法的货币，在法律上没有偿还性和强制性等货币特点。例如，中国人民银行就明确表示，比特币不属于货币，而是属于可以交易的虚拟商品，国内所有金融机构不得开展与比特币相关的业务。尽管如此，比特币和以太币等加密货币还是具备了流通性、可支付性、稀缺性等货币的基本特征。

传统货币的拥有人可以把货币存在银行里，或者通过钞票等实物形式用于支付。加密货币则不同，基本上由持有人自己保管账号的数字信息。在比特币的系统里面，账号是由椭圆曲线数字签名算法（ECDSA）中的公钥，经过哈希变换，再加上校验码而生成的一串数字，通常是一个 33 或 34 位的 Base58 编码字符，例如：16UwLL9Risc3QfPqBUvKofHmBQ7wMtjvM。使用 Base58 编码字符的原因是尽量减少账号的长度，把大小写的英文字母以及数字（除去容易混淆的数字 0、大写 O、大写 I 和小写 l）都用来表示数字。这个账号在比特币里称作地址（address），用户接收他人发来的比特币，或者发送比特币给他人，都需要用到这个地址（为方便读者理解，本章将在大多数情况下将账号和

地址作为同义词）。所以，比特币里的账号（地址），就像具有门牌号码的信箱，任何人只要知道这个门牌号码，都可以往信箱里面转入比特币。另一方面，信箱的主人有把特殊的钥匙，就是与 ECDSA 公钥对应的私钥，只有使用这个私钥，才可以打开信箱并取走（花费）存放在里面的比特币。由此可见，私钥就是比特币主人需要自己保管和保密的账号信息。需要指出的是，若忘记了传统银行账号密码，可以找银行重置账号密码，而在比特币体系里面，一旦丢失私钥，则没有办法可以“重置”私钥，将也打不开信箱拿走账号里的比特币，这是比特币无中心化的机制决定的。

既然私钥那么重要，管理比特币等加密货币资产实质上就是私钥的保管和使用。最理想的管理办法是既能保障资产安全又能很方便地使用资产（支付或花费），然而现实中这两个目标往往不可兼得。例如，一种简便的方法就是把私钥保存在电脑或手机里，需要支付比特币的时候，可以通过软件用私钥签名并发送交易。但这种方式的缺点也很明显，如果电脑或手机丢失，或者私钥文件损坏，将无法操作相关账号，账号中的比特币也等同于丢失了。还有一种情形，如果电脑等设备被黑客攻破或感染病毒，保存的私钥可被复制，进而账号中的加密货币也会被盗走。

为了应对设备丢失或文件损坏的情况，可把私钥文件备份到其他设备上，如 U 盘、移动硬盘等，然后把备份设备存放在安全的地方，遇到设备故障可以恢复私钥。对于黑客、病毒等外来攻击，因为电脑和手机需要联网才能支付加密货币，理论上就没法彻底避免私钥被盗的可能，需要依靠其他的方式来应对。简单地说，就是“分仓存放”，使用多个账号地址来存放加密货币。在频繁使用的在线账号里存放少量的加密货币，在电脑等设备上要保存这些账号的私钥。其余加密货币都转到离线的账号中，所谓离线账号，就是该账号的私钥不在任何的连线设备上存放。每当在线账号里的钱增加到一定程度时，可以将其转移到离线账号中保存，而且每次转移可以使用不同的离线账号。比特币的离线账号还有个优点，就是往账号地址里面转入比特币时，账号无需联网，这样就大大增加了安全度，如图 9-1 所示。

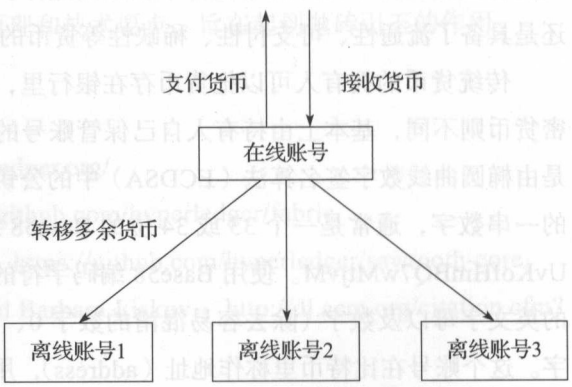


图 9-1 在线账号和离线账号的关系

如果账号很多的话，用户难以记住众多账号地址和私钥，因此需要有效的管理方法管理账号。钱包（wallet）就是管理加密货币账号及其交易的软

件。钱包软件一般具有良好的操作界面，协助用户方便地完成各种货币交易。此外，钱包还能够创建新的账号地址（包括公钥和私钥对），从而使用户的交易更具有匿名性。需要指出的是，钱包软件创建新地址的时候是用随机数的方式生成的，并不需要联网校验新地址是否已被他人使用。一方面，是因为有很多账号从没有发生过交易，不能从公共账本中查找到；另一方面，是因为地址相同的概率实在太低，以比特币的私钥为例，它是一个 256 位的二进制随机数，两个用户产生相同地址的概率只有约 $1/10^{78}$，与连续 8 次中六合彩头奖的概率相当（每次中头奖概率约 $1/10^{9}$），可见担心用户地址会冲突实在是杞人忧天了。

早期的钱包软件需要把整个账本的数据下载到本地，以便获取账号和交易的相关信息。账本记录了比特币系统从创始日开始的所有交易，数据量有几十个 GB，并且其大小与日俱增。目前用户多使用优化过的轻钱包，只需下载区块数据的头部信息（header）和用户账号相关的交易，这样使得数据量减少到原来的 1/1000 左右。

随着云计算的流行，又出现了在线钱包的 SaaS（Software as a Service）云服务。用户不用安装任何软件在电脑或手机里，通过网络即可直接访问云端的钱包应用。其好处就是使用方便，对于大多数没有网络安全知识的用户来说，采用专业的钱包服务商是个比较好的选择。当然，其中也有一定的风险，就是账号的私钥可能会被钱包服务商保存或得到，所以用户需要信任服务商会好好保管私钥并且不会泄露私钥。用户可以参照上文提到的“分仓存放”原理，把在线钱包里暂不使用的货币转移到离线账号中，以提高安全性。

9.2　加密货币的交易方式

加密货币多数是通过挖矿的方式产生的，矿工手里积累了大量的加密货币，矿工们需要支付挖矿的硬件、电费等成本，客观上有出售加密货币的需求。其他另一部分人则要使用加密货币来完成一些任务，如以太币可以用来执行智能合约，比特币可支付某些转账的交易费等。这样，既有供给方，又有需求方，加密货币就具备了交易的基础。

比特币是最早的加密货币，比特币的早期交易是由人工撮合的 OTC（Over The Counter）方式，通过交易员（中间人）实现买卖双方交换的目的。交易员支付主权货币（如美元、人民币等），从比特币持有人手里买入比特币，再寻找并卖给合适的买家，赚取主权货币的差价。交易员还可以通过撮合双方直接交易从中收取佣金。即使在电子交易网站盛行的今天，OTC 模式还没有消失，主要是部分用户更信任的人（交易员），而

不是第三方的网站。而且在交易数量大的时候，OTC 方式可避免成交价格被大抛单砸下的损失，从而获得比交易网站更高的售价。

虽然线下的 OTC 交易模式会继续存在，但是必须有交易员参与人工的方式也制约了交易数量的扩大。因此，当前加密货币交易主要依靠各种交易所来完成。交易所采用电脑自动撮合买家和卖家的方式，能够 24 小时不间断地交易。一般可以交易多种加密货币，如比特币、以太币、莱特币等，不仅可实现各种电子货币之间互相兑换（主要是和比特币兑换），还可把电子货币和主权货币进行兑换。

交易所加密货币的交易流程的简化模型如图 9-2 所示。步骤描述如下：

1）买家把资金（如人民币等法定货币）转入交易所的账号，实质上是从买家的银行账号转账资金给交易所的银行账号，交易所在其系统里给买家的资产增加一笔同等金额法定货币；

2）卖家把加密货币（如比特币）从自己的地址转移到交易所的地址，这个交易将反映在加密货币公开的区块链账本中，交易所在其系统中给卖家增加相应的加密货币资产；

3）交易所的计算机系统根据买卖双方的报价自动撮合交易，如果成交，卖家将从资产中转移加密货币到买家的资产中，同时将买家相应的法定货币转移到卖家的资产中；

4）买家可把买到的加密货币提取到自己拥有的地址中，交易所在加密货币的账本中发送加密货币到买家的地址，并扣减买家在交易所中的加密货币资产；

5）卖家可把得到的法定货币提取到自己的银行账号中，交易所在银行发起转账交易，把法定货币发送给卖家，并扣减卖家在交易所中的法定货币资产。

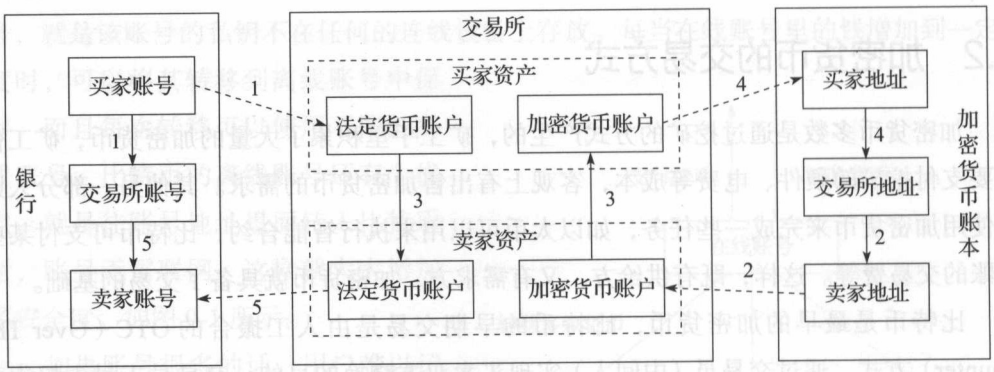


图 9-2 加密货币交易流程的简化模型

从上述过程可以看出，参与交易的双方需要把原来自己拥有的资产转移到交易平台中，然后才能进行交易。在交易所中产生的各种买卖交易，其实只是在交易所的记账系

统里面进行的（如上述步骤 3），从外界的银行或者加密货币账本系统来看，这些交易都不存在。因此，对交易者来说，交易所的信用或可靠性就是很重要的选择因素。在已有的案例中，有的交易所是庞氏骗局，有的则因受到黑客攻击导致资产被盗而破产，用户的资产损失最终都无法挽回。可见，在交易所平台进行加密货币交易时，在便捷之余，还要承受一定的风险。比较明智的做法是，当不做交易的时候，用户把资产从交易所中转到自己控制的账号地址（数字货币）或银行（法定货币）中，以避免交易所带来的风险。

9.3　匿名性和隐私性

加密货币经常被宣传的优点之一就是匿名性（anonymity），实际情况是怎样的呢？我们以比特币为例子分析一下。

1. 不具备真正的匿名性

用户是通过地址来使用比特币的，账本中的交易只记录了地址和比特币数量，从交易中无法直接得到用户的真实信息，这就是人们常说的比特币匿名性。如果比特币仅仅是在虚拟数字世界里面流转，确实比较难发现用户的身份信息。可是，当用户一旦用比特币和现实世界的事物发生联系，用户的信息就可以关联上某个地址。例如，用户在商店用比特币购买一杯可乐，店员就可以把比特币支付地址和该用户的相貌、性别等信息关联起来。如果用比特币网购商品，那么买家的姓名、地址、电话等信息就会透露给卖家。还有不少人在网站、视频、文章中留下自己的比特币地址来接收捐赠或收款，殊不知这样就泄露了自己和比特币地址的关系。有的国家法律要求，从事比特币等业务的机构（如比特币交易所）需要了解用户真实信息（Know Your Customer，KYC），以符合反洗钱（Anti-Money Laundering，AML）等一系列监管规定。这样，用户的信息也被政府或相关机构所掌握。一些学术研究的结果也印证了比特币的匿名性难以保证。例如，瑞士苏黎世联邦理工学院和德国 NEC 欧洲实验室的学者们研究发现，即使采用了隐私保护措施，40% 的比特币用户身份信息仍能够被识别出来的。因此，比特币地址不具备真正的匿名性，更准确地说，应是假名性（pseudonymity），就像用户在网络论坛上使用的代号一样，虽然不知到用户是谁，但用户一言一行都可关联到这个代号上。

2. 隐私性无法保障

除了匿名性之外，每个地址的隐私性几乎是无法保证的。因为比特币的每一笔交易都会公开记录在区块链账本上，任何人都可以查阅。只要通过分析每个地址发生过的交

易，就可以发现很多的账号之间的关系。我们来看看下面的例子。

比特币的交易是记录各个地址之间转移货币的数量，通常有一个或多个地址作为输入，还有一个或多个地址作为输出。在通常使用的习惯下，同一个交易中的输入地址一般可以认为是由同一人所有。例如，张三要支付 3 个比特币给李四购买一台电脑，张三有两个地址，各有 2 个比特币，这个交易如图 9-3 所示。

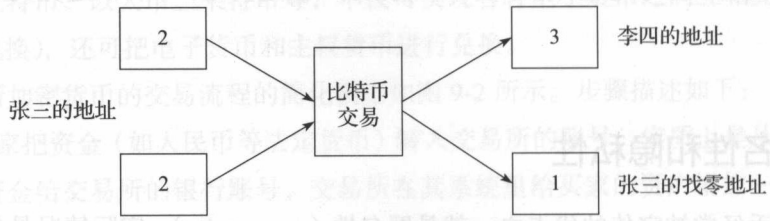


图 9-3　比特币交易的输入和输出

这个交易会广播到比特币所有节点上并持久化下来。其他人通过比特币区块链上公开的数据，可以推测出这笔交易的两个输入地址很可能属于同一个人。然后按照同样的方法，可以分别追踪两个输入地址发生的其他交易，从而发现更多的地址相关性。另外，配合其他信息，从交易的输出地址也可能分析出更多结论。例如，有的钱包软件总是把找零地址作为输出的第一个地址，这样可推断出第一个输出地址和输入地址属于同一人所有。另一些钱包软件总是创建全新的找零地址，那么他人通过查找账本，可把未发生过交易的地址归结为找零地址。

从上面例子看到，比特币是完全透明的系统，通过各种数据挖掘技术，可以发现很多地址的相互关系。从积极的方面说，政府监管机构可以从中发现洗钱、行贿等犯罪的蛛丝马迹；而从消极的方面说，用户的隐私却无从保障，就好比将每笔交易都用假名在微信、微博等社交媒体上发布，一旦假名的真实身份泄漏，所有交易将暴露在公众的眼皮底下。

估计没有哪个用户希望自己的交易情况被别人知道得一清二楚，保护隐私成为比特币系统中重要的问题。为此，Blockstream 的开发者 Gregory Maxwell 提出了名为“CoinJoin”的方案。

3. CoinJoin 方案

CoinJoin 的原理比较简单，就是把不同用户的多个交易合并成一个交易，如图 9-4 所示。外人从这种混淆过的交易中无法假定输入地址属于同一人所有，也不能确定货币的流向。用户可以进行多次 CoinJoin 操作，进一步隐藏交易的关系。

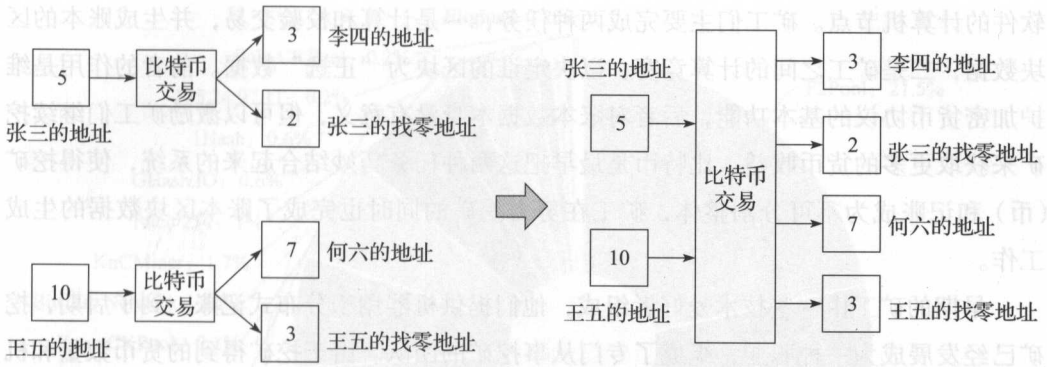


图 9-4　用 CoinJoin 来混淆交易的输入和输出

CoinJoin 不需要改变比特币的协议，实施起来比较容易。已经实现的方式较多是采用中间服务器，需要 CoinJoin 的用户可在服务器上登记，由服务器聚合多个用户请求生成一个大交易，经过各个用户签名后再发布到比特币网络上。尽管这种方式能够混合用户交易，但缺点是中间服务器可以掌握所有用户的输入输出地址，对于该服务器来说用户毫无隐私可言。如果该服务器被攻破，用户的隐私也将无从保证。一种潜在的解决办法是用户对服务器加密，屏蔽服务器对交易细节的了解，目前这种方法还在研究中。另一种解决办法是去除中间服务器，采用去中心化的方式来组织用户的交易。已经有若干个这类实现，如 Coinjumble、Coinmux、CoinJoiner 等，但是都没有被广泛使用。

经过 CoinJoin 处理过的交易是否就能够保护用户的隐私呢？ Blockchain.info 公司提供称为 SharedCoin 的 CoinJoin 服务，有研究人员发现它并不能完全掩盖交易地址，通过工具分析数据特征，还是可以发现原来交易的输入和输出地址。其中一个原因是各个用户输入的比特币数量都不尽相同，因此，根据输入等于输出（可相差少量交易费）原理，可以把原交易识别出来。一种改进办法是要求参与混合交易的用户使用相同的输入金额，这样可减少用于区分交易的特征。不足之处是需要更多时间“凑齐”一组用户来合并交易。

匿名性和隐私性是加密货币领域一直在研究的问题，用户在使用的时候，应该充分了解这些特点，以采取适合自己的隐私保护策略。

9.4　矿池算力集中的问题

以公共区块链分布式账本为基础的加密货币，如比特币、以太币等，需要靠分布在世界各地的矿工（miner）不停地运作来维持系统功能。所谓矿工，就是运行加密货币

软件的计算机节点。矿工们主要完成两种任务，一是计算和校验交易，并生成账本的区块数据，二是矿工之间的计算竞赛，以决定谁的区块为“正选”数据。前者的作用是维护加密货币协议的基本功能，后者对账本数据本身没有意义，但可以激励矿工们继续挖矿来获取更多的货币收益。比特币是最早把这两种任务巧妙结合起来的系统，使得挖矿（币）和记账成为不可分割整体，矿工在努力挖矿的同时也完成了账本区块数据的生成工作。

早期的矿工由一些技术爱好者组成，他们提供机器用于分布式记账。到了后期，挖矿已经发展成为一种职业，组成了专门从事挖矿的团队。由于挖矿得到的货币数量和机器的运算能力大小成正比，因此从概率上看，采用越快速的硬件，在所有矿工中算力的占比越高，就能够获得越多的货币。矿工们为了获得更高的收益，彼此之间在算力上进行较量，算力低的矿工会因挖不到币而逐渐被淘汰出局。从参与的硬件上看，最开始矿工们采用通用的 CPU（中央处理器）来挖矿。后来大家发现 GPU（图形处理器）能够提高并行计算能力和吞吐量，效率更高，于是纷纷采用 GPU 挖矿。再后来，出现了专门为挖矿设计的集成电路（Application Specific Integrated Circuit，ASIC）芯片，目前已经成为挖矿行业主流的硬件设备。

随着全网算力的不断增加，单打独斗的小矿工已经没有规模优势，挖到加密货币的时间非常不确定，运气不好的话可能要几年才能挖到一个有效区块，有点像买彩票中奖一样[⊖]。为了使收入更加平稳，矿工们可以组成矿池（Mining Pool），由矿池管理者统一分派挖矿的计算任务，挖到的币都归矿池管理者所有。矿池管理者根据各个矿工贡献的算力比例，定期分配挖矿的收入。矿池已经成为加密货币区块链网络算力的主要来源，零散的矿工由于经济上不占优势，基本上已退出了挖矿的行列。

矿池能够给矿工带来相对稳定的收入，但是也带来了新的问题。矿池把原来分散的算力集中起来统一管理，这违背了区块链的去中心化原则，在矿池规模不断增加的过程中，有的矿池在全网的算力达到了相当大的比例，甚至排前几位的矿池的算力总和可以超过全网的 51%。图 9-5 是 2016 年 7 月的比特币算力分布情况，可以看到前 3 大矿池的算力已经大于 51%（这 3 个矿池都在中国）。从理论上说，如果能够控制整个网络达到或超过 51% 以上的算力，就可以控制区块链的记账权。这样比特币等加密货币依赖的分布式记账方式将被破坏，同一个货币可以多次使用（即重复花费，也叫双花，double spend），这样，信用体系将不复存在，加密货币体系将被彻底摧毁。

⊖ 实际上比特币等加密货币的 PoW 共识算法，因其具有一定的随机性，有时也称作彩票算法。

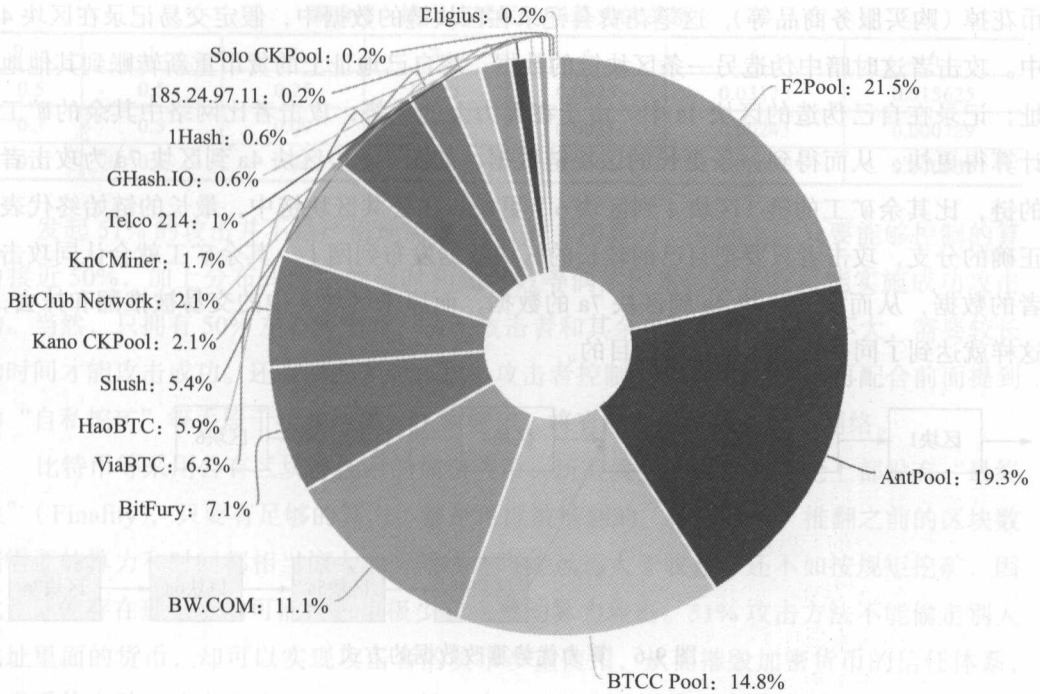


图 9-5　比特币矿池算力分布

51% 算力的攻击问题一直是加密货币体系中的“达摩克利斯之剑”，包括矿池在内的所有参与者都会非常小心地避免出现这种情况。一方面，对矿池来说，增大规模对自己有利，可以挖到更多的币，而且在一些特殊事件上有较多的话语权，例如在社区讨论加密货币软件重大变更的时候，由于最终的软件采用决定权在矿工，因此，矿池的规模越大，表决权就越大。另一方面，如果少数几个矿池的算力总和超过或接近 51%，整个系统就不再是去中心化的系统，将会引发用户对加密货币的信任危机。所以，规模太大的矿池，通常会自觉地停止接收新成员，以避免与系统玉石俱焚。加盟矿池的矿工，也会尽量选择分散算力的矿池，避免一家独大的情形出现。

9.5　51% 攻击问题

矿池算力集中产生 51% 攻击问题，准确来说，应当是 50%+ 问题，从原理上看，只要能控制全网 50% 以上算力，攻击者将可以修改账本和阻止他人挖矿，从而威胁到整个系统安全。那么拥有 50% 以上的算力是怎样劫持区块链的数据的呢？一种方法是通过分叉（forking）的方式。如图 9-6 所示，正常的矿工在区块链上挖矿，攻击者把自己的货

币花掉（购买服务商品等），这笔花费会记录在区块链的数据中，假定交易记录在区块 4 中。攻击者这时暗中伪造另一条区块链的数据，将自己地址上的货币重新转账到其他地址，记录在自己伪造的区块 4a 中。由于在算力上的优势，攻击者比网络中其余的矿工计算得更快，从而得到一条更长的区块链数据，如图 9-6 中区块 4a 到区块 7a 为攻击者的链，比其余矿工的链（区块 4 到区块 6）更长。在公共区块链中，最长的链始终代表正确的分支，攻击者只要把自己的较长的区块数据发布到网上，其余矿工就会认同攻击者的数据，从而接受区块 4a 到区块 7a 的数据，而原来区块 4 中的交易被推翻和抹去，这样就达到了同一货币双重花费的目的。

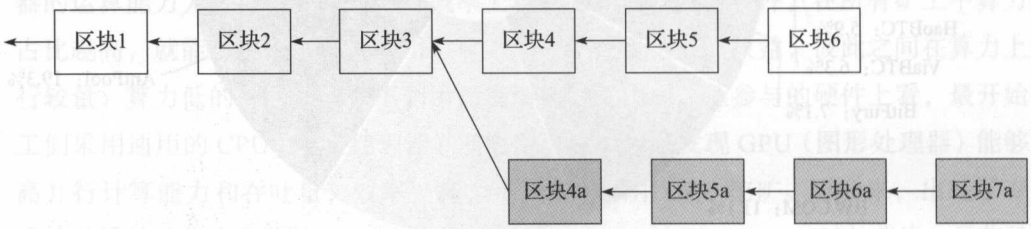


图 9-6　算力优势篡改数据的方式

从上述原理可以看到，攻击者要想更改账本数据，需要有一条比其他人更长的区块链数据。但是要实现这个目的，并不一定要拥有超过 50% 的算力。如，某个矿工运气特别好，挖到两个连续的区块。他可以先隐藏这两个区块，等到其他矿工挖出一个区块数据后，他再广播自己的两个区块，就可使得他人挖到的区块失效（因为长链占优）。这种做法称为“自私挖矿”（selfish mining），这就白白浪费了其他矿工的算力，而自私者获得了更多的收益。

那么矿工连续挖到两个区块的概率是多少呢？如果某矿工的算力占全网的比例为 p，他连续算出 n 个区块的概率就是 p^n。表 9-1 列出了部分 p 和 n 对概率的影响，当 p=0.1（10%）时，连续挖到 2 个区块的概率为 1%；如果 p=0.5（50%）时，连续挖到两个区块的概率为 25%，这已是较大概率的事件了。从比特币的历史上看，确实发生过矿池连续挖到多个区块的事情。在比特币系统中，6 个区块之前的数据被认为是相对可靠和难以推翻的。可是在 2014 年，大矿池 Ghash.io 挖到 6 个连续的区块，具备了逆转之前交易的可能，引发了社区对比特币安全性的担忧。该矿池当时的规模在 50% 上下，算出 6 个连续区块的概率约为 1.5%（参见表 9-1）。概率虽小，但日积月累不停地挖矿，使得小概率事件也很可能发生。为此，Ghash.io 也不得不承诺把规模控制在 39.99% 以下。

表 9-1　挖出连续区块和算力的关系

P | n	1	2	3	4	5	6
0.5	0.5	0.25	0.125	0.0625	0.03125	0.015625
0.3	0.3	0.09	0.027	0.0081	0.00243	0.000729
0.1	0.1	0.01	0.001	0.0001	0.00001	0.000001

发起 51% 的攻击并不是严格地需要 51% 的全网算力，实际上，只要能够控制的算力接近 50%，加上分布式网络的延时和随机数等偶然因素，还是有可能实施成功攻击的。当然，只拥有 50% 左右算力时，由于攻击者和其余矿工的算力相差不大，需要较长的时间才能攻击成功。还有研究表明，如果攻击者控制了 33% 的算力，再配合前面提到的“自私挖矿”等手段干扰其他矿工正常挖矿，将有能力最终控制整个网络。

比特币等采用公有区块链技术的加密货币，所有发生的交易在理论上都没有“最终性”（Finality），只要有足够的算力，都是可以被推翻的。在实际中，推翻之前的区块数据需要的算力和时间都相当惊人，所需成本可能远远大于收益，还不如按规矩挖矿，因此，即使存在理论上的可能性，也极少有人发动算力攻击。51% 攻击方法不能偷走别人地址里面的货币，却可以实现攻击者的货币多重使用，从而摧毁加密货币的信任体系，使得币价大跌，攻击者或许可以通过做空货币的方式获利。鉴于 51% 攻击的破坏性强，加密货币的参与者（如开发者、用户、矿工等）都会密切监视可能存在的攻击，并随时进行防备。

9.6　去中心化的自治组织

9.6.1　去中心化的自治组织简介

去中心化的自治组织 DAO（Decentralized Autonomous Organization）是随着数字加密货币和区块链技术的普及而流行起来的概念。去中心化的组织最早出现在美国作家奥里·布莱福曼（Ori Brafman）在 2007 年出版《海星和蜘蛛》一书中，描述如下：蜘蛛是中心化（细胞）组织的例子，如果把它的头切掉后（整个组织）就无法生存了。海星则是由彼此对等（无中心）的一堆细胞组成的，海星撕下的每只触手都可成长为完整的海星。海星和蜘蛛分别代表现实世界中去中心化和中心化的两种组织。海星型组织在遇到挫折和冲突被分解的时候，其组织将变成更小的去中心化组织，继续发挥作用；而蜘蛛型组织在首脑被割掉之后，将无法继续运作。相比之下，海星型去中心化运作的组织将具有强大的生命力。

奥里的去中心化组织概念更多的是指人类组织的互相协作方式。近年来，由于去中心化的区块链技术发展，特别是智能合约的出现，演变出以机器代替人类执行事务的去中心化的自治组织（DAO）。如果组织是商业公司的形式，有时候也称作去中心化的自治公司 DAC（Decentralized Autonomous Corporation）。DAO 目前没有严格的定义，但通常是指一个去中心化组织用计算机程序来描述和实现所有业务规则，并且无需人类的管理和干预，也能够自我运作。DAO 之所以能成为现实并生根发芽，离不开它最重要的技术基础——区块链。

区块链本质上是去中心化的分布式数据库，数据复制到整个网络的所有节点上，靠多数节点间的共识来保持数据的一致性。只要节点数目足够多、分布足够分散，就没有一个人或组织能够控制和决定所有节点的行为。有了区块链这个去中心化的基础平台，自治组织就可以定义各种组织规则，然后用计算机程序来表述并且在区块链网络上运行，这种区块链上的程序通常称为“智能合约”（Smart Contract）。智能合约中的“合约”是指用计算机代码确定下来的逻辑规则，发布到区块链网络后，形成一种不可更改的公开契约，所有人都可以监督契约的内容及其执行。而智能合约的“智能”，是指计算机程序能够根据各种不同的情况（即智能合约的调用参数以及节点的状态），做出不同的响应。综上所述，DAO 就是运行在区块链网络上的、体现组织规则的智能合约。区块链网络保证了其去中心化特性，并通过铁面无私的机器执行智能合约来确保其自治性。

DAO 是人类史上前所未有的组织形式，成员甚至可以匿名参与并且不分国界。作为一个社会或经济实体，目前各国均无法界定它的法律地位，甚至在某些国家可能会被认定为非法。另外，由于 DAO 所有的组织规则都蕴含在代码当中，所谓“代码即法律”（Code is Law），评判是非曲直完全依据代码所表达的意义，在很多场合，特别是与现实社会中的道德准则发生碰撞时，将难以辨析 DAO 所承担的社会责任。

目前已经有一些 DAO 创建出来了，如 Dash、DigixDAO 和 The DAO。其中最具有代表性和争议性的是 The DAO，前前后后发生了许多足以记入史册的事件，引发了区块链系统中的部分问题，如智能合约的漏洞处理、软分叉、硬分叉、重放攻击等。让我们一起来看看 The DAO 事件引起的争议和带给人们的思考。

9.6.2 The DAO 项目

The DAO 是由 Slock.it 公司发起的一个众筹项目。Slock.it 原先计划通过物联网和区块链技术，提供智能锁等设备，把人们之间的租赁关系用去中心化的方式建立起来，如租房、租自行车等。他们打算采用 DAO 的模式来运作这套租赁系统，于是，几个创

始人就开发了在以太坊平台上的智能合约，可以众筹到所需的资金（以太币形式），并且可以按照一定的规则运作（如出租自行车等），在取得经营上的收益后，可以回馈众筹参与者。在开发的过程中，他们发现这个智能合约的框架可以被其他类似的 DAO 项目重新使用，所以他们改变了主意，决定创建 The DAO，意为“DAO 之母”（The Mother of all DAOs）。创立 The DAO 的想法，由于思路独特新颖，前途光明，吸引了众多人的关注，众筹额一路攀升，在 2016 年 5 月结束的时候，筹集的资金竟达到了 1.6 亿美元，一举成为史上最大的众筹项目。

那么，拥有万众瞩目的 The DAO 到底是怎么运作的呢？总体上说，The DAO 有如下几个特点：

1）The DAO 本质上是个风险投资基金（Venture Capital，VC），通过以太坊筹集到的资金会锁定在智能合约中，没有人能够单独动用这笔钱。更重要的是，该组织只存在于虚拟的数字世界中，不受任何政府监管约束，无国界。且资金是加密数字货币的以太币（Ether）形式，其行为由智能合约中的代码来主导（因此称作“自治”）。

2）每个参与众筹的人按照出资数额，获得相应的 DAO 代币（token），具有审查项目和投票表决的权利。从这点上说，DAO 代币有点像股票，众筹参与者是股东。代币还有另一个作用，就是持有人有权提出投资项目的议案，供 The DAO 审核。

3）投资议案由全体代币持有人投票表决，每个代币一票。如果议案得到需要的票数支持，相应的款项会划给该投资项目。在传统基金中，投资策略是由经验丰富的基金经理等专业人士制定的。而在 The DAO 中，决策来自于“众智”（the wisdom of the crowd）。众智的概念最早可以溯源到亚里士多德关于政治的论述，其原理是：综合许多人的智慧，可以做出比某个专家更好的结论。现实生活中这类例子很多，如陪审团、维基百科的编制、《百万富翁》游戏中的询问观众等。当然，这里抛开了纯粹的“众智”，而加入了出资额的权重。

4）投资项目的收益会按照一定规则回馈众筹参与人（股东）。当然，并不是每个项目都有收益，或者能够盈利，而且一般来说会是风险较高的项目，可以靠分散投资来降低风险。

需要指出的是，Slock.it 即便是 The DAO 代码的缔造者，也无法从规则或控制权上获得先天优势，他们也需要向 The DAO 提出申请来获取资金。因为 The DAO 一旦启动运作，就由刚正不阿的机器和预设的程序代码不改初衷地执行，这也是人们信赖 The DAO 的原因。

本来 The DAO 筹集完资金就开始运作了，但却发生了一件惊天动地的事情——The

DAO 的智能合约出现了漏洞（即程序的 bug），被黑客利用，并把超过 30% 资金的 360 多万个以太币（约 5000 万美元）转移到黑客控制的子 DAO 账号中，并且在理论上黑客可以把所有资金偷光。既然已经知道被攻击了，为什么不能赶紧采取措施阻止剩余资金继续被盗呢？个中原理和 The DAO 的去中心化机制有直接关系。智能合约的代码一旦发布出去，就无法更改，除非事先留有可应急“刹车”的后门。但是 The DAO 的目标是完全自治的去中心化组织，绝对不允许有这样的后门。因此，即使发现了代码中的 bug，却无法制止这种行为，所有人都只能眼睁睁地看着钱被偷走而无能为力。

9.6.3 代码漏洞分析

本节分析 The DAO 代码的缺陷，完整的代码可在 Github 网站上下载（https://github.com/TheDAO/DAO-1.0）。根据 The DAO 白皮书的设计，代码中 splitDAO 函数的运行可以分裂出一个小规模的 DAO。这个函数的本意是要保护投票中处于弱势地位的少数派，防止他们被多数派通过投票的方式合法剥削。通过分裂 DAO，给予他们一个拒绝的机制，同时仍然确保他们可以获取分裂前进行的对外资助产生的可能收益。

splitDAO 会创建 childDAO（如果不存在的话），将分裂者拥有的以太币转入 childDAO 中，支付任何已产生的报酬（Reward），然后返回。攻击者利用了这一目前唯一可行的提取以太币的机制，创建 childDAO 并将以太币持续地转入给定的账户。

攻击者分析了 DAO.sol 代码，发现 splitDAO 函数在递归发送模式上存在漏洞：该函数只是在最后才修改用户的结余和交易总额，如果能够在返回 splitDAO 处理这些运算之前，进行多次以太币操作调用，就能通过多次递归来持续转移以太币到其他账户。

splitDAO 函数中与漏洞相关的代码段如下，我们将逐一分析。

```
function splitDAO(
    uint _proposalID,
    address _newCurator
) noEther onlyTokenholders returns (bool _success) {
    ...
    // 转移以太币并赋予新的代币，注意，这是首先运行的代码
    uint fundsToBeMoved =
        (balances[msg.sender] * p.splitData[0].splitBalance) /
        p.splitData[0].totalSupply;
    if (p.splitData[0].newDAO.createTokenProxy.value(fundsToBeMoved)(msg.
        sender) == false) // 这是攻击者希望多次运行的代码
        throw;
    ...
    // 使用 DAO 代币
```

```
    Transfer(msg.sender, 0, balances[msg.sender]);
    withdrawRewardFor(msg.sender); //好了，可以拿到他的报酬了
    //注意在进行以下操作前，上一行的操作是致命的
    totalSupply -= balances[msg.sender]; //而这在最后运行
    balances[msg.sender] = 0; //这也在最后运行
    paidOut[msg.sender] = 0;
    return true;
}
```

先看看 splitDAO 函数的定义，函数的修饰符表示，其调用者不能为以太币（ether）账户，而应该为代币持有者（Tokenholder）账户。

```
function splitDAO(
uint _proposalID,
address _newCurator
) noEther onlyTokenholders returns (bool _success) {
```

再看下一段代码，可以用于以太币转账：计算向一个特定的调用者转账多少以太币，之后调用 createTokenProxy 函数。createTokenProxy 源代码在 TokenCreation.sol 中，它会将代币从待分裂的父 DAO 转移到子 DAO 中。基本上攻击者就是重复调用 createTokenProxy 函数来获得更多的代币并转移到子 DAO 中。

```
//转移以太币并赋予新的代币，注意，这是首先运行的代码
uint fundsToBeMoved =
    (balances[msg.sender] * p.splitData[0].splitBalance) /
    p.splitData[0].totalSupply;
if (p.splitData[0].newDAO.createTokenProxy.value(fundsToBeMoved)(msg.
    sender) == false) //这是攻击者希望多次运行的代码
    throw;
```

接着，最后发送以太币的代码出现了。

```
    //使用 DAO 代币
    Transfer(msg.sender, 0, balances[msg.sender]);
    withdrawRewardFor(msg.sender); //好了，可以拿到他的报酬了
    //注意在进行以下操作前，上一行的操作是致命的
    totalSupply -= balances[msg.sender]; //而这在最后运行
    balances[msg.sender] = 0; //这也在最后运行
    paidOut[msg.sender] = 0;
    return true;
}
```

上面这段代码中，先调用 withdrawRewardFor 函数，然后设置 totalSupply、balances 以及 paidOut 变量。这使得合约中，为 msg.sender 记录的以太币余额归零、扣减以太币

总量 totalSupply 等操作都发生在将以太币发给 msg.sender 之后。很明显，这种方式会出现参数竞争（Race to Empty）攻击，如果设法在 balances 或 paidOut 变量更新前调用 withdrawRewardFor 多次（如利用递归方式），可能导致 withdrawRewardFor 会不断地成功调用。实际上，withdrawRewardFor 是有漏洞的。

```
function withdrawRewardFor(address _account) noEther internal returns (bool _
    success) {
        if ((balanceOf(_account) * rewardAccount.accumulatedInput()) /
            totalSupply < paidOut[_account])
            throw;

        uint reward =
            (balanceOf(_account) * rewardAccount.accumulatedInput()) /
                totalSupply - paidOut[_account];
        if (!rewardAccount.payOut(_account, reward))
            throw;
        paidOut[_account] += reward;
        return true;
    }
```

这段 withdrawRewardFor 代码不长，一个判断加上一个 payOut 调用，payOut 的值在 rewardAccount.payOut 调用之后设置，这不是良好的编程习惯。

当 reward 设置后，rewardAccount.payOut 函数被调用，rewardAccount 是 Managed-Account 类型的合约，其 payOut 函数定义如下：

```
function payOut(address _recipient, uint _amount) returns (bool) {
        if (msg.sender != owner || msg.value > 0 || (payOwnerOnly && _
            recipient != owner))
            throw;
        if (_recipient.call.value(_amount)()) {
            PayOut(_recipient, _amount);
            return true;
        } else {
            return false;
        }
    }
```

_recipient.call.value()() 调用时，没有 gas 数量要求。语句 recipient.call.value(_amount)() 向 _recipient 账户转账 _amount 个 Wei，调用默认会使用当前剩余的所有 gas，于是无论该语句调用多少次，gas 限制只取决于攻击者当时可用的 gas 数量。

于是攻击开始了。

1）在以太坊区块链上创建一个钱包合约，包含默认的函数调用 splitDAO 若干次，但次数不要太多，以免 gas 不够。过程中还需要记录当前调用深度以控制堆栈使用情况。

2）使用钱包合约的接收人地址创建分裂申请。

3）等待若干时间让分裂完成。

4）调用 splitDAO。

这时，调用栈类似如下（假设钱包仅调用两次，实际上次数更多）。

```
splitDao
    withdrawRewardFor
        payOut
            recipient.call.value()()
                splitDao
                    withdrawRewardFor
                        payOut
                            recipient.call.value()()
```

之后，攻击者就可以等待以太币源源不断地转入了。

从代码看，本次攻击成功的因素如下：

（1）应用代码顺序问题

以太币余额扣减和以太币转账这两步操作的顺序有误。应先扣减以太币的余额再转账以太币，因为以太币的余额检查作为转账以太币的先决条件，要求以太币的余额状况必须能够及时反映最新状况。在问题代码实现中，尽管最深的递归返回并成功扣减黑客的以太币余额，但此时对黑客以太币余额的扣减已经无济于事，因为其上各层递归调用中余额检查都已成功，不会再有机会判断最新余额了。

（2）不受限制地执行未知代码问题

虽然黑客当前利用了 solidity 提供的匿名 fallback 函数，但这种对未知代码的执行原则上可以发生在更多场景下，因为合约之间的消息传递完全类似于面向对象程序开发中的方法调用，而提供接口等待回调是设计模式中常见的方法，所以完全有可能执行一个未知的普通函数。

9.6.4　解决方案

由于 The DAO 的智能合约代码存在缺陷，发布后无法更改，因此，依靠 The DAO 自身去解决黑客的问题已经无望，唯一的办法是求助于 The DAO 的运行平台以太坊。由于问题影响巨大，甚至影响了以太坊的声誉和运作，包括创始人 Vitalik 在内的以太坊

核心团队迅速行动，全力阻止黑客的攻击。幸运的是，The DAO 设计中有缓冲期，黑客控制的以太币要在 27 天后才可拿走，这留给了以太坊团队足够的应对时间。他们设计的方案分两步：

1）采用软分叉（soft fork）技术，锁定 The DAO 及其子 DAO 的账号，不允许发生任何交易，相当于冻结了黑客的以太币，使其无法出售获利。软分叉实际上是在以太坊软件中增加约束性的规则（例如不允许某账号转账等），好处是不影响任何以太坊上已发生的交易，无需回滚区块链的数据。

2）在软分叉的基础上，实施硬分叉（hard fork），把 The DAO 智能合约中的以太币（包括黑客控制部分）转到一个新的智能合约当中，以便退回给众筹参与者。这种改动需要永久性改动以太坊的协议规则，相当于直接修改 The DAO 和黑客的账户余额，有违区块链数据不可变更的原则。

尽管上述方案在技术上可以挽回的黑客造成的损失，却引起了社区激烈的争论，参与者有程序员、研究员、分析师、投资人、律师等各方人士。辩论的焦点回到了老子《道德经》中的经典六字悖论："道可道，非常道"，也就是求证天地万物最崇高的道应该如何描述。方案支持者认为，通过更新以太坊代码，使得黑客的阴谋没有得逞，伸张了正义，维护了平台的公平性，也维护了"道"。虽然这次问题没出在以太坊平台本身，但 The DAO 若失败，对以太坊也是个很难抹去的污点。支持者绝大多数是 The DAO 或以太币的持有者。反对者则无法接受以太坊团队为单个应用（The DAO）买单而修改平台规则的行为，认为这样违反了区块链不可篡改数据、去中心化的根本原则，将大大损害以太坊的公信力和公正性，恰恰是失道之举，完全可以说是得不偿失。两派争论的焦点是到底是保护 The DAO 用户的利益重要，还是维护以太坊的去中心化的公正性更重要？

从本质上看，人类发明的各种软件、工具、机器等系统都是为完成某个目标而设计的。区块链作为去中心化的系统，是为了实现在没有中间人的情况下，保证数据的真实性和完成特定业务逻辑的方法。当区块链应用（如 The DAO）出现故障，甚至造成投资人重大损失的时候，只为盲目地坚守去中心化和数据完整性而不做任何纠正，应该是舍本逐末的做法。此次修改规则，以太坊团队只是提出建议，执行还需要多数参与者（主要是参与记账的矿工）认可才有效。由此看来，这提议还是个去中心化的决定，只不过达成共识的是理性的人类，而不是麻木的机器。之前比特币就出现过类似故障，也是通过社区认同的方式，用硬分叉方式解决了问题。因此，从社区的反馈来看，大部分人还是支持分叉方案的实施的。

9.6.5　软分叉和硬分叉的影响

区块链软件的升级并不是一件简单的事情，因为全网所有节点的软件更新有先有后，甚至有的节点不会更新，需要考虑不同版本软件之间互相协作的兼容性，特别是更改系统协议的软硬分叉升级，更要重视新旧协议切换带来的问题。

在软分叉升级中，因为新规则变得更严格了，新节点（升级后的节点）会拒绝某些交易的记账，而旧节点（未升级的节点）仍认可这些交易。如果新节点占多数，那么产生的区块数据就是按照新规则的；此时，旧节点所生成的区块，由于算力小，区块高度小，而且是短链而最终被放弃。可见，尽管在过渡期间有可能存在临时性的分叉，当多数节点都切换成新版本软件后，软分叉一般不会产生永久性分叉的链。

在硬分叉中，按照新规则产生的区块只被新节点认可，旧节点则不认为新区块合法，因此，旧节点一直用旧规则来组织自己的链，所以，无论新旧节点数目的比例如何，一定会产生两条链。若大多数节点都升级到新版本，旧链能否成活要看有多少算力的支持。

以太坊为解决 The DAO 的问题，首先实施了软分叉方案。简单地说，在软分叉升级后，对每个以太坊上的交易，验证节点（矿工）都会检查是否与 The DAO 智能合约及其子 DAO 的地址相关。如果是则拒绝这个交易，从而锁定 The DAO（包括黑客在内）的所有资金。这个逻辑实现本身并没有问题，但是却没有收取执行交易的燃料费（gas）。当初以太坊设计的 gas 有两个作用：支付矿工的交易费和增加潜在攻击者的成本。取消交易燃料费后，导致了非常严重的后果：以太坊网络成为了 DoS（Denial of Service）的攻击目标（如同我国在法定节假日高速公路免费通行造成大塞车的情况一样），攻击者可以零成本发起大量无效交易，使得整个网络彻底瘫痪。因为这个漏洞，各个节点被迫回滚了软件版本，软分叉方案宣告失败！

以太坊社区随后又设计和实施了硬分叉方案，在该方案中，软件会在第 1880000 区块把 The DAO 合约及其所有子 DAO 合约归入一个列表 L 中。在第 1920000 区块中，设计一个非常规的状态变更，强行把 L 中所有地址的余额都转到一个特定的退款合约地址 C，这个合约唯一的功能就是把众筹人的 DAO 币换回以太币。经过充分准备，硬分叉终于成功切换，以太坊上大部分算力都切换到了支持硬分叉的新版本。

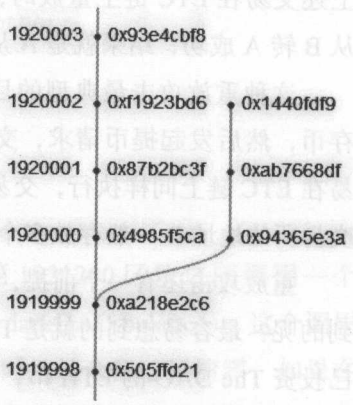


图 9-7　硬分叉前后区块的情况

如图 9-7 所示是硬分叉前后时刻第 1920000 区块的

情况，左侧数字为区块编号，右侧十六进制数代表区块的哈希值。直线代表分叉后新链，大多数矿工都升级了软件并在这条链上记账（挖矿）；曲线分支代表原（旧）链，少数没有升级的矿工，依然停留在旧链上记账。分叉后，新链出块的速度大于旧链，说明大部分算力都已切换到新链上了。

之前在比特币等区块链上也出现过硬分叉的情况，当时绝大多数矿工都切换到新链上，旧链也就自动消亡了。可这次以太坊硬分叉却不同，由于社区存在分歧，部分矿工还继续维护着旧链，因此旧链并未消失。后来有交易所介入并支持旧链上的以太币交易，使得旧链上的算力陡增，以太坊由此分裂为新旧两条具有相同历史账本的区块链。

9.6.6 重放攻击

以太坊同时存在新旧两条链，为了区分，新链仍称为以太坊（货币代号：ETH），旧链则称为经典以太坊（货币代号：ETC，Ethereum Classic）。两链的软件代码相同（除了涉及 The DAO 的部分），历史账本一样（分叉前），地址的私钥也一样，交易广播到两个网上都是合法的，这就引出了重放攻击（Replay Attack）的问题。如，在新链上发送货币的交易，同样会广播到旧链上，并且交易还能成功（下面一段会讲到交易失败的例外），可被利用来作为攻击手段。

假设分叉前地址 A 有 10 个 ETH，地址 B 有 50 个 ETH。分叉后在 ETH 链上从地址 A 转 10 个 ETH 到地址 B 上，再把 10 个 ETH 从 B 转回 A，地址 A 最终还是拥有 10 个 ETH（扣除少量 gas 损耗），地址 B 的余额不变。上述交易在 ETC 链上也广播一次（重放），同样也会执行从 A 转 10 个 ETC 到 B，再从 B 转 10 个 ETC 到 A，一切都还正常。但是，如果攻击者设法使得在 ETH 链上 A 的余额为 10 个 ETH，在 ETC 链上余额为 0，上述交易在 ETC 链上重放时，从 A 转 B 操作会因余额不足失败；而因 B 的余额足够，从 B 转 A 成功，结果就是 A 从 B 取走了 10 个 ETC。

这种重放攻击最典型的目标就是交易所，攻击者从自己的地址 A 向交易所地址 B 存币，然后发起提币请求，交易所在 ETH 链上确认交易并把币转到 A，殊不知这个交易在 ETC 链上同样执行，交易所的 ETC 就被人提走了。为什么目标是交易所呢？因为交易所的地址 B 一般存放多个客户的 ETC 币，余额较大，这样攻击容易得手。

重放攻击还有一个前提，就是需要有个地址 A 在两条链中的余额不同，这是怎么做到的呢？最容易想到的就是 The DAO 的众筹人，他们在硬分叉后在 ETH 链上可取回自己投资 The DAO 的 ETH 币，可是这些 ETH 在对应的 ETC 链中依然存放在 The DAO 的合约中，因为分叉时，ETC 和 ETH 的差别就是在 The DAO 合约的余额上，如图 9-8 所

示。The DAO 众筹人取回自己的 ETH 后，就拥有了这样一个地址，在新旧链中余额不同。下文还会介绍另一种错开地址在新旧链余额的方法。

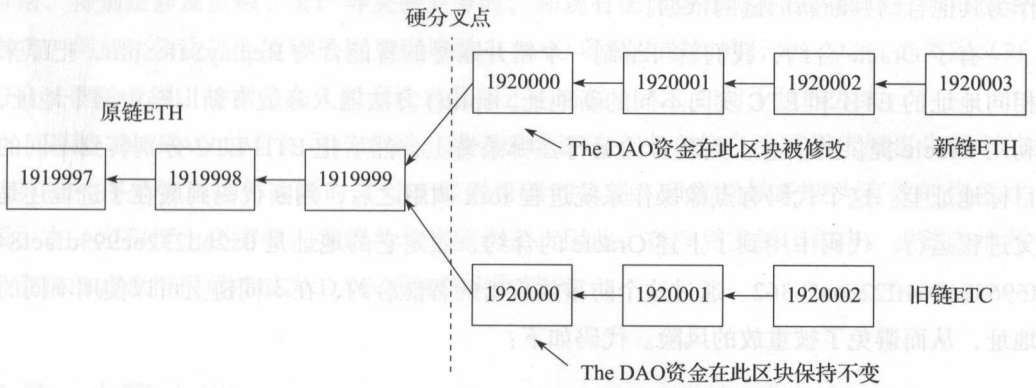


图 9-8　硬分叉前后区块数据对比

ETH 和 ETC 同源的两条区块链并存，交易互相交错，一时之间带来使用上的混乱，在任何一条链上做交易，都要考虑在另一条链上是否有重放影响。最好的办法还是把 ETH 和 ETC 存放在不同的地址上，从而互不影响。为了实现这个目标，可在分叉前部署一种叫作 Oracle（预言家）的智能合约，合约名称为 AMIOnTheFork，代码如下：

```
contract AmIOnTheFork {
    bool public forked = false;
    address constant darkDAO = 0x304a554a310c7e546dfe434669c62820b7d83490;
    // 在分叉后的 1920000 ~ 1920001 区块之间检查分叉条件
    // 大约是在 2016-07-20 12:00:00 UTC 到 2016-07-20 17:00:00 UTC 之间
    // 此后，状态将锁定在变量 forked 中
    function update() {
        if (block.number >= 1920000 && block.number <= 1921200) {
            forked = darkDAO.balance < 3600000 ether;
        }
    }
    function() {
        throw;
    }
}
```

这个智能合约要在硬分叉后的第 1920000 区块和第 1921200 区块之间调用一个 update() 方法，该方法会检查 The DAO 黑客的地址余额（darkDAO.balance）。这个调用会同时被新旧两链执行，如果是在新链的节点上执行，此时地址余额已经清零；如果在旧链上执行，黑客地址余额必定大于 3600000。这样，该合约把自己处于哪条链的状态

区分出来，并记录在变量 forked 中，forked 作为智能合约的状态持久化到区块链上。需要注意的是，在新旧链中，forked 的值是不同的，这样埋伏在区块链中的 Oracle 也就可作为其他合约判断新旧链的依据。

有了 Oracle 合约，我们就可编制一个错开账号的智能合约 ReplaySafeSplit，把原来相同地址的 ETH 和 ETC 发向不同的新地址。split() 方法输入参数有新旧链中两个地址，利用 Oracle 提供的信息，判断合约运行在哪条链上，然后把 ETH/ETC 分别转到不同的目标地址上。这个代码有点像操作系统进程 fork 调用之后，判断代码到底在子进程还是父进程运行。代码中用到了上述 Oracle 的合约，假定它的地址是 0x2bd2326c993dfaef84f696526064ff22eba5b362。通过这个防重放攻击的智能合约，在不同链上可以使用不同的地址，从而避免了被重放的风险。代码如下：

```
contract ReplaySafeSplit {
    // 调用 fork 状态的 oracle 智能合约地址
    AmIOnTheFork amIOnTheFork = AmIOnTheFork(0x2bd2326c993dfaef84f69652606
        4ff22eba5b362);

    // 根据处于哪条区块链，判断以太币发向哪个账号
    function split(address targetFork, address targetNoFork) returns(bool) {
        if (amIOnTheFork.forked() && targetFork.send(msg.value)) {
            return true;
        } else if (!amIOnTheFork.forked() && targetNoFork.send(msg.value)) {
            return true;
        }
        throw;
    }
    // Reject value transfers.
    function() {
        throw;
    }
}
```

The DAO 项目最终以失败告终，但仍是人类史上具有开拓意义的重要尝试，它试图建立完全由计算机程序控制的去中心化自治实体，实现公正、透明和开放的运作模式。当然，丰满的理想也会遇上骨感的现实，DAO 超前的理念落地还需要面对很多实际问题。

首先，当自治的智能合约出现问题时，应该采用怎样的方式解决呢？一种方案是用另一个智能合约来解决产生的纠纷，但这样又陷入一个死循环中，因为新的智能合约可能也存在问题。另外一种方案是采用众人表决的形式，这要求 DAO 预留解决纷争投票

的接口，在需要的时候可以启用。当然，此方案被认为自治性受到干预。

其次，区块链技术被主流社会体系认可的过程也许比较漫长，当 DAO 尚不能完全自治，特别是涉及金融、资产等关键业务时，和现有法律道德体系还需要良好的衔接的方式。例如法律法规要能够适用于区块链上 DAO 等新生事物的行为，DAO 也要定义好符合法律的规则，特别是在代码无法自我纠正的时候，人工如何干预的方式。

通过 The DAO 的事件可以看到，基于区块链的去中心化自治组织依照事先编制的计算机程序来管理运作。同时，对于代码可能的出错，目前尚缺乏较为有效的恢复手段，在一定程度上还需要人类的参与才能纠正。因此，在今后发展过程中，机器自治和人类辅助监管将很可能成为 DAO 系统的主要模式。

9.7　本章小结

本章描述了一些区块链和加密货币领域的常见问题，如怎样安全保存和使用数字加密货币，如何保持匿名性和隐私性，集中式矿池带来的问题，以及去中心化自治组织 DAO 的运作原理和面临的挑战。可以看到，作为新生事物的区块链和加密货币，进入实用阶段还需要解决不少问题，这也是留待今后区块链研究发展的课题。

第10章

区块链应用案例分析

本章介绍在区块链上两个应用系统的设计和实现原理。其中，闪电网络是为了解决比特币系统交易吞吐量低的问题，它通过巧妙的离线交易形式，支持大量的高频微支付交易。另一个系统是ODIN，利用区块链的去中心化和不可篡改等特点，提供DNS的服务功能。

10.1 闪电网络

10.1.1 闪电网络简介

比特币是现今最成熟的数字货币系统，无需任何中间人，用户可在比特币的网络里转移货币，实现对商品和服务的支付能力。但作为实用的支付系统，比特币还存在着一些缺陷。如，比特币平均每秒只能处理大约7笔交易，全年的吞吐量约2.2亿笔，无法满足一个城市的基本需求；每笔交易要等到1小时后才能基本确认；对微支付（micropayment）来说，交易费用可能太高。虽然有些方案可以提高比特币的性能，效果却不太显著。

闪电网络（Lightning Network）由Joseph Poon和Tadge Dryja首先提出，被认为是比特币创立以来最重要的革新。它利用了比特币的安全特性，在线下提供高速的实时交易处理能力。用户既可通过点对点的直接支付方式，也可以通过网络路由的方式实现间接支付。闪电网络并没有发明新的加密形式，也没有使用新奇的比特币脚本，却巧妙地

实现了离线支付的功能。目前闪电网络还在实现阶段，相信不久将进入实用阶段。在其他加密货币（如以太币）系统中，也按照类似的原理实现了离线支付方案，如雷电网络（Raiden Network）。本节主要介绍闪电网络的工作原理。

10.1.2　支付通道的创建

在闪电网络提出之前，支付通道（Payment Channel）的概念就已经出现。支付通道可以实现某些特定场合的离线交易，但仅仅支持单向支付，有点像预付费的购物卡，只能不断地把钱从用户转向商户。闪电网络对支付通道进行了扩展，实现了无需信任第三方的双向支付通道。

双向支付通道可由参与的双方共同发起一个交易来创建，实质上就是双方往一个多重签名（multisig）地址存入一定数量的比特币。假定 Alice 打算给 Bob 发送 1 个比特币，Alice 可直接在比特币的网络上广播一个交易，从 Alice 的地址转向 Bob 的地址。但是考虑到两人经常有来往交易，他们决定向同一个地址 X 各转入 3 个比特币，X 拥有 6 个比特币的未花费输出（UTXO）。由于该地址是 2 分之 2（2-of-2）的多重签名地址，因此需要两人同时签名才可以使用地址中的比特币。这个交易称为支付通道的“首次交易”（opening transaction），如图 10-1 所示。在向比特币网络广播“首次交易”之前，Alice 和 Bob 各自确定一个随机数作为密码（secret），然后对密码进行哈希运算，把哈希值（hash）告诉对方（密码各自保留）。在后面可以看到，这个哈希值和密码还可以用来取消过期的离线交易。

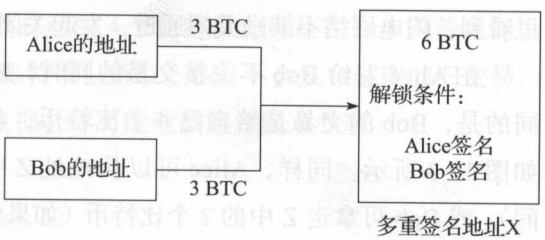


图 10-1　支付通道的首次交易

接下来就是 Alice 和 Bob 各自创建离线交易，更改通道的离线余额。Alice 创建的交易称作“承诺交易”（Commitment Transaction），是把首次交易中的输出地址 X 作为输入地址，发送 2 个比特币给自己，4 个比特币给一个新的多重签名地址 Y，如图 10-2 所示。这个多重签名地址 Y 很特别，有两种花费方式：一种方法是 Bob 可以单独广播交易来解锁拿到 4 个比特币，但需要再等待 1000 个出块时间（从交易在区块链上确认后算起），这是用比特币的 CSV（CheckSequenceVerify）锁来实现的；另一种方法是由 Alice 单独广播交易来解锁，前提是 Alice 得到和 Bob 的哈希值对应的密码。一般情况下，Alice 是没有 Bob 的密码的，后面我们会看到在取消该交易的时候，Alice 可以得到 Bob 的密码。

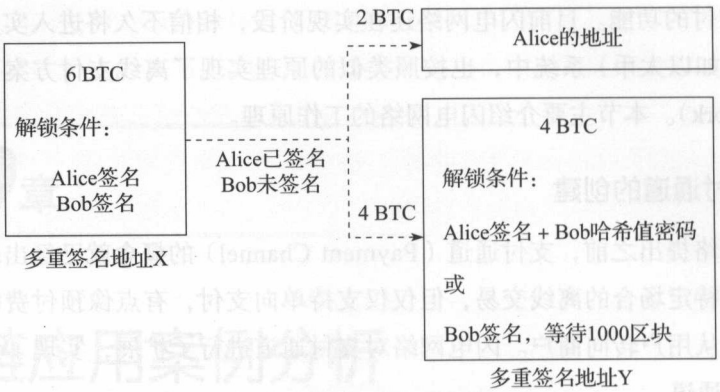


图 10-2 Alice 发给 Bob 的不完整交易

Alice 在上述交易中签名，由于输入地址 X 需要多重签名，这个交易还需要 Bob 的签名才能生效。Alice 把这个缺少 Bob 签名的不完整交易线下发给 Bob（即不通过比特币网络），这点正是闪电网络设计精妙之处：Bob 可随时在不完整交易中签名，然后广播到比特币网络上去确认记账，也就是说，Bob 任何时候能够确保自己获得 4 个比特币（需要等待 1000 个出块时间），而 Alice 可即时得到 2 个比特币。当然，Bob 广播这个交易的同时，地址 X 的余额被用完，支付通道也就自动关闭了（相当于清算了两人的账目）。这里看到，闪电网络不鼓励关闭通道，发起关闭交易的一方，会比对方晚些获得比特币。

在 Alice 发给 Bob 不完整交易的同时，Bob 也给 Alice 发送类似的不完整交易。不同的是，Bob 的交易是给自己 4 个比特币，将 2 个比特币发给另一个多重签名地址 Z，如图 10-3 所示。同样，Alice 可以从地址 Z 中拿走 2 个比特币（需等待 1000 个出块时间），或 Bob 可拿走 Z 中的 2 个比特币（如果他知道 Alice 的密码）。

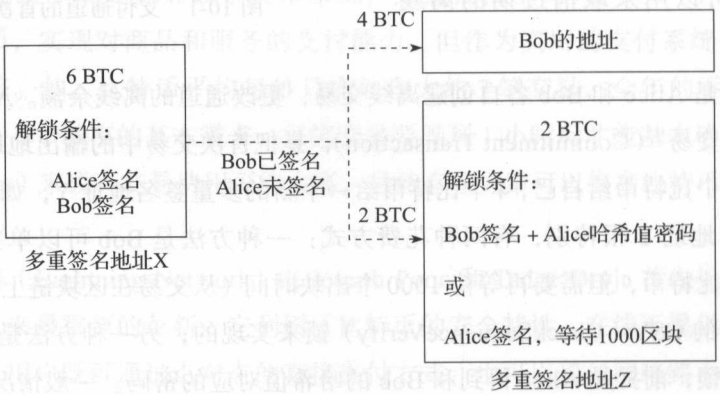


图 10-3 Bob 发给 Alice 的不完整交易

在 Alice 和 Bob 交换了不完整交易以及密码的哈希值之后，他们共同签名的首次交易就可以广播到比特币网络上去确认，支付通道正式打开。上面已经提到，任何一方都可以从通道中退出并结算比特币余额，但是发起方会比另一方晚得到比特币。在不需要结算的时候，他们各自保存对方发来的不完整交易。

10.1.3　支付通道的更新

过了一段时间，当 Bob 希望给 Alice 发送一个比特币这时候，他们可以在线下完成这笔交易，具体操作和上节的承诺交易一样。Alice 发给 Bob 一个不完整交易，给自己 3 个比特币，给 Bob 3 个比特币；Bob 也同样发给 Alice 一个类似的不完整交易，给自己 3 个比特币，给 Alice 3 个比特币。和上次不同的是，Alice 和 Bob 各自要使用新的密码，并和对方交换新密码的哈希值。这样操作之后，通道中 Alice 和 Bob 的比特币数目将从 2 : 4 更新为 3 : 3，实现了 BoB 转 1 个比特币给 Alice 的交易。

此刻，Alice 和 Bob 各持有对方的 2 个不完整交易，为了防止任何一方广播已经过期的交易，闪电网络在更新支付通道余额的时候，要求双方把上一轮交易中各自的密码发给对方，这实际上等同于作废了上一轮交易中双方持有的不完整交易。例如，Bob 如果把旧的交易签名发到比特币网络上（因为对 Bob 更有利），Alice 可立刻得到 2 个比特币，Bob 需要等待 1000 个出块时间才能拿到自己的 4 个比特币。可是在 Bob 等待之时，Alice 可用 Bob 的密码立刻取走 Bob 的 4 个比特币！所以，Bob 不会发布过期的旧交易。同理，Alice 也不会发布过期的旧交易，两人都会尽可能地按照规则来办事。

10.1.4　支付网络的构建

支付通道解决了两个用户在线下直接交易的问题。但如果两个用户之间没有建立支付通道，他们怎样交易呢？闪电网络设计了一种中间人方案：若 Alice 希望发送 1 个比特币给 Carol，她们之间没有直接的支付通道，但她们分别和 Bob 有支付通道，于是可以通过 Bob 作为中间人来支付。Alice 把 1 个比特币给 Bob，Bob 再把 1 个比特币给 Carole。

这种支付方法比较容易理解，在实施中需要解决各方间无信任的问题，例如，Bob 需要确保他支付 1 个比特币给 Carol 后，会收到 Alice 发来的 1 个比特币。闪电网络采用了 HTLC（Hashed Timelocked Contract，哈希时间锁合约），来保证交易能够畅通进行。

在 HTLC 中，Alice 不会直接发送 1 个比特币给 Bob，而是发送 1 个比特币给一个多重签名地址 Q，如图 10-4 所示。地址 Q 有两种解锁方式：用 Bob 的签名加上其他人

设置的一个密码；或者 Alice 用本人签名解锁，但需要等待一段时间，例如 30 天。这样的方式相当于在 30 天内，Bob 如果得到密码，就可以用签名拿走地址 Q 中的 1 个比特币；如果 Bob 在 30 天内没有提供密码，被认为超时（timeout），Alice 可以从地址 Q 中拿回自己的 1 个比特币。

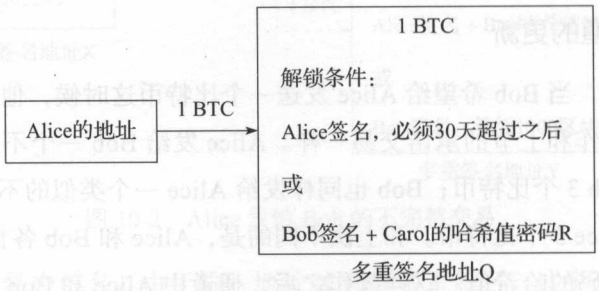


图 10-4 Alice 和 Bob 的 HTLC 交易

同样的，Bob 和 Carol 也设立这样的 HTLC，并且满足两个要求：采用和 Alice-Bob 的 HTLC 相同的密码，以及稍短一些的超时设置，如 29 天。Bob-Carol 的 HTLC 超时要比 Alice-Bob 的 HTLC 要早，目的是确保在 Bob 支付了 Carol 之后，可以从 Alice 取回相应的比特币，而不是因超时 Alice 取回比特币。因此，在 HTLC 中，采用了 CLTV（CheckLockTimeVerify）锁，来确保在未来特定时间点（如某个区块高度）超时解锁。

从 Alice 到 Carol 的整个支付过程如图 10-5 所示。首先 Carol 随机生成一个密码 R，再把 R 进行哈希运算后得到 H(R)，然后将 H(R) 发送给 Alice。Alice 用哈希值 H(R) 作为 CLTV 锁，创建和 Bob 的 HTLC 合同，包含 1 个比特币，有效期 30 天。Bob 用同样的 H(R) 哈希值打开一个和 Carol 的 HTLC 合同，也包含一个比特币，有效期 29 天。

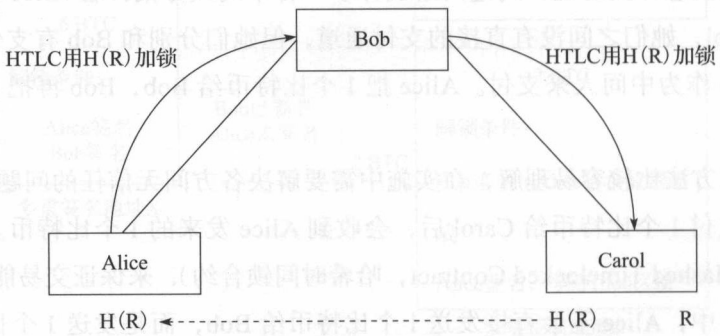


图 10-5 Alice、Bob 和 Carol 的 HTLC

在 29 天内，Carol 可以提供密码 R，来解锁 Bob 的 HTLC，从而得到 1 个比特币。Bob 也看到了密码 R，他可用 R 去解锁 Alice 的 HTLC，也得到了 1 个比特币，这样就完成了从 Alice 到 Carol 的支付，如图 10-6 所示。

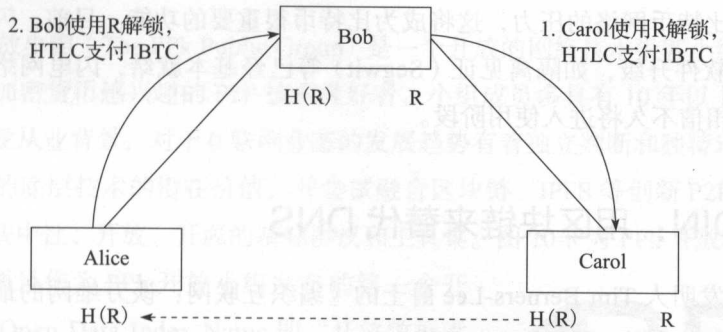


图 10-6　Carol 用 R 解锁 Bob 的 HTLC 后，Bob 也可用 R 解锁 Alice 的 HTLC

如果在 29 天内 Carol 没有拿走比特币，因为 Bob 的 HTLC 已经超时，Bob 可以取回自己的比特币；因为 Bob 没有 Carol 的密码 R，无法取走 Alice 的 HTLC 中的比特币，Alice 等到第 30 天，也能取回自己的比特币，如图 10-7 所示。

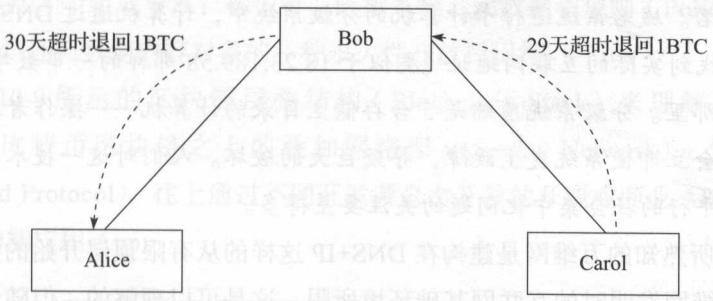


图 10-7　超时后，Bob 先赎回比特币，Alice 随后赎回比特币

闪电网络还可以通过多个中间节点实现转账，像互联网的多节点路由一样，只要两个用户之间存在一条通路，就可以按照上述原理进行支付。

10.1.5　支付通道的关闭

上文已经提到，Alice 或 Bob 两人都可随时把最新的不完整交易签名后发布的网络上，从而关闭支付通道，发起关闭的一方需要等待较多的时间才能拿到自己的比特币。当然，双方也可以协商共同发布一个正常的交易，按照大家认可的余额比例来转移首次交易中的输出，双方都无须等待时间，离线的所有不完整交易也同时作废。

10.1.6 小结

闪电网络实现了在离线环境下提供比特币交易的方式，在支付通道打开后，参与方可离线发生任意数量的交易，而无须广播到比特币的网络上，从而大大提高了交易速度，也减轻了比特币网络的压力，这将成为比特币很重要的功能。目前，闪电网络依赖的一些必要的软件升级，如隔离见证（Segwit）等已经基本就绪，闪电网络本身的实现也即将完成，相信不久将进入使用阶段。

10.2 ODIN：用区块链来替代 DNS

在万维网发明人 Tim Berners-Lee 爵士的《编织互联网：谈万维网的最初设计和最终命运》一书中有这样一段话：

虽然万维网是在一种松散的状态下成长的，但事实上它有一个中心的致命弱点，通过这个弱点可以击垮或控制它。当用 http://www.lcs.mit.edu/foo 这样的 URI（Universal Resource Identifier，统一资源标识符）来寻找一个网页时，客户机会先检查第一个字符串，就像通常的情况那样，当它是 http 时，它就知道 www.lcs.mit.edu 这个部分是万维网服务器的域名。域名系统运行于计算机的分级系统中，计算机通过 DNS（域名解析系统）搜索它来找到实际的互联网地址（类似于 18.23.189.58 那样的一串数字），然后数据包就被发送到那里。分级系统顶端是 5 台存储主目录的计算机——操作者在其中任何一台上的错误都会立即使系统发生故障，导致巨大的破坏。人们对这一技术弱点本身的关注要比对与之平行的社会集中化问题的关注要差得多。

目前大众所熟知的万维网是建构在 DNS+IP 这样的从有限顶层开始的多层级授权基础上的，受万维网发明时的互联网基础环境所限，这是可以理解的，但随着互联网的快速发展，商业化资本的涌入和逐利已将开放、失控的万维网引向封闭、受控状态，这已经限制了万维网的最初设计：让人们可以平等自主地分享信息。

同时作者 Tim Berners-Lee 爵士认为“将类似公钥加密技术引入万维网对于个人表达信任是根本性的”，“万维网之所以能成功，就是因为任何人都可以有制作链接的能力，使它能展现现实生活中的各种信息和关系。密码学并不经常体现万维网上信任的原因是，还不存在一个万维网式的分权式的基础结构”。

横空出世的比特币所带来的区块链技术，正是 Tim Berners-Lee 当初所设想的“能充分将密码学带入万维网的分权式的基础结构”。基于区块链技术可以定义出开放、中性的命名标识体系，自主、唯一、安全、持久，进一步结合更多的网络技术就可以定义

出新的传输协议，相比原有的 DNS+IP 体系，它可以更好地适配和推动万维网的进化。下文介绍的 PPk ODIN 开源项目正是基于这样的理念在做尝试。

10.2.1　ODIN 简介

PPk 开放小组（The PPk Public Group）是一个开放的网络技术极客小组，集合了一群对比特币等加密货币感兴趣的 P2P 技术爱好者，小组成员多具有 10 年以上通信和互联网行业技术研发从业背景，对于互联网业态的发展趋势有着独立判断和独特理念，关注以区块链为代表的底层技术的潜在价值，并尝试融合区块链、IPFS 等创新 P2P 技术，来定义一个实现一些中性、开放、开源的基础协议和工具集。图 10-8 为 PPk 开放小组图标。

图 10-8　PPk 开放小组图标

ODIN 项目作为 PPk 开放小组发布的第一个开源项目，是 Open Data Index Name 即“开放数据索引命名标识”的缩写。广义上，ODIN 是指在网络环境下标识和交换数据内容索引的一种开放性系统，它遵从 URI 规范，并为基于数字加密货币区块链（BlockChain）的自主开放、安全可信的数据内容管理和知识产权管理提供了一个可扩展的框架。它包括 4 个组成要素：标识符、解析系统、元数据和规则（Policies）。狭义上，ODIN 是指标识任何数据内容对象的一种永久性开放标识符。

参考图 10-9 所示的区块链层叠结构（Blockchain Stack）来理解，PPk 的 ODIN 项目定位于比特币区块链之上的叠加网络层（Overlay Network）+ 分布式协议层（Decentralized Protocol），往上通过不同开发者自由开发的开源或商业 API，可以支持更多具体业务功能应用 App。

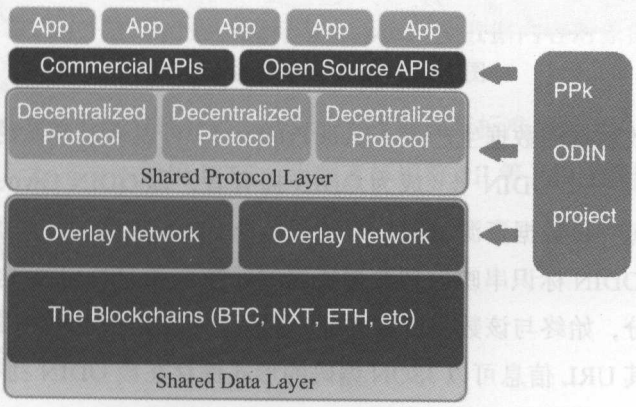


图 10-9　ODIN 开源项目示意图

ODIN 可以被形象地理解为“数据时代的自主域名”，是基于比特币区块链定义并可扩展兼容更多区块链的完全开放的、去中心化的命名标识体系，相比传统的 DNS 域名它有更多创新特性，可以很好地被应用到大数据、智能设备和物联网等新兴领域。

10.2.2 实现功能

PPk 开放小组定义并发起建立了一套完全开放、开源的 ODIN 运行机制，如图 10-10 所示。它包括基础技术和扩展应用方面，其中，基础技术是一个基于区块链的分布式数据内容索引命名与标识系统，它保证数据内容的自主开放性、安全性、唯一性、永久性和可扩展性。

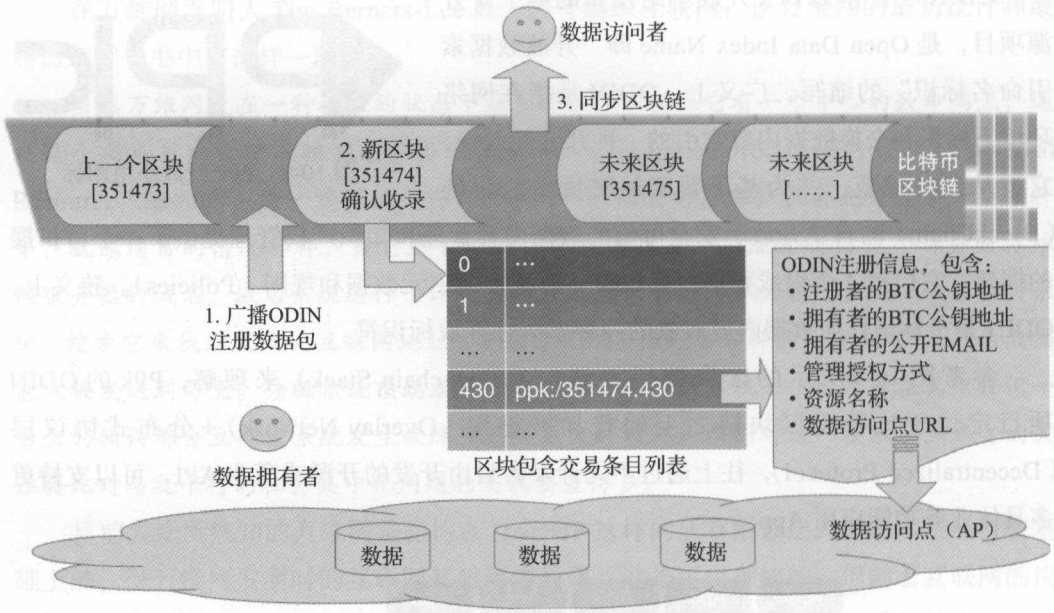


图 10-10 ODIN 基础运行机制

每个有意开放数据的数据生产者（Data Producer）可以通过开源的 ODIN 注册客户端来自主注册获得一个 ODIN 号（成为 ODIN 拥有者，即 ODIN Owner）。以此为前缀可以为其开放的每一份数据资源编制一个包含本身 ODIN 前缀的、增加了后缀的 ODIN 标识串，并将该 ODIN 标识串映射到数据资源的元数据和 URL 上，这样 ODIN 就成为数据资源的一部分，始终与该数字资源共存。然后，已被开放的这些数据资源的 ODIN 记录、元数据及其 URL 信息可以 JSON 编码的形式保存在该 ODIN 拥有者的数据库内，这些被集中存贮起来的资源就形成一个 ODIN 资源标识库。

当用户根据 ODIN 标识串寻找一个数据资源或有关这一资源的相关信息时，查询请求就会通过开源的 ODIN 解析库在区块链上进行定位，然后传送到该 ODIN 拥有者所登记的访问点（Access Point）上进行解析，并得到该数据资源的元数据描述和实际数据 URL 链接。

ODIN 拥有者可以完全开放数据资源访问权，也可以通过适当的自定义机制让用户获取数据资源访问权，如通过订购、资源传递、按浏览付费或者预印本付费等方式获得。

有条件的个体可以自主提供代理注册服务（称为 ODIN 注册者，ODIN Register），ODIN 注册者缺省拥有所代理注册 ODIN 的维护私钥，并有权对 ODIN 的注册登记信息进行配置更新，在经 ODIN Register 和 ODIN Owner 双方确认同意的情况下，可以将注册者转移更新为由 ODIN Owner 拥有私钥的比特币地址，这时 ODIN Owner 将获得完全自主的权利，等同于自主注册。

10.2.3　主要特点

基于区块链技术定义的 ODIN 标识体系具有以下显著特性。

- 自主性：ODIN 标识符基于去中心化的区块链技术，由申请者自主生成并管理，其生成和管理规则是完全开放的，没有中心化的控制机构。除了拥有管理密钥的申请者之外，其他组织和个人都无权控制和篡改。
- 安全性：每一个 ODIN 标识符的拥有者都对应拥有一对非对称加密技术的公私钥，可以通过私钥对自主发布的数据内容进行签名，接收数据内容的个体可以通过公钥进行验证，以确保收到的数据是来源可信和未被篡改的。
- 唯一性：ODIN 标识符能对任何数据内容对象（如文本、图片、声音、数据、影像、软件等）的开放访问索引进行唯一标识，使数据内容对象能被人们准确地识别和提取。
- 兼容性：ODIN 可以与现有的一些标识符（如国际标准书号 ISBN、国际标准刊号 ISSN、国际标准文本代码 ISTC、出版者标识符 PII 等）相互兼容。

10.2.4　ODIN 标识编码格式

1. 一级基础 ODIN 标识

一级 ODIN 的标准结构式为：

```
ppk:[BTC_BLOCK_SN].[BTC_TRANS_INDEX]/[DSS]
```

ODIN 分为命名体系、前缀和后缀 3 部分，用冒号、斜杠分开。前缀中又以小圆点分为两部分 [BTC_BLOCK_SN] 为该 ODIN 在比特币区块链上的登记记录所在区块的数字流水号（由比特币网络决定）。[BTC_TRANS_INDEX] 为该 ODIN 在比特币区块链上的登记记录所在区块内的具体存储位置的阿拉伯数字编号（从 0 开始寻址）。后缀 [DSS]（Data Suffix String）是由 ODIN 拥有者自选并自行给出的该 ODIN 标识所对应具体数据内容定位标识，需要自主确保具有唯一性。

[DSS] 的命名方案如下：

```
RESOURCE_ID#[DATA_BLOCK_SN.CHUNK_INDEX]
```

或者

```
#[DATA_CHUNK_INDEX]
```

其中：

- RESOURCE_ID 为对应的资源标识，由所属 ODIN 标识拥有者来定义，可以是流水编号，也可以是唯一取值的字符串，需自行保证能与标准结构式区分开，且不能包含“#”和“/”字符。
- DATA_BLOCK_SN 为对应的内容区块编号（从 1 开始，对于文件 File 可理解为版本号，对于动态数据流 Stream 可以理解为顺序产生的数据包）。
- CHUNK_INDEX 为对应的子数据块在该区块内部的索引编号（从 0 开始）。
- DATA_CHUNK_INDEX 则表示所有区块的子数据块记录的第几个子块（从 0 开始）。

注意："#" 字符及其后续部分可省略，缺省表示对应最新内容区块的第一个子数据块。

下面几例都是符合定义的一级 ODIN 的合法编码：

```
ppk:351474.430/
ppk:351474.430/#
ppk:351474.430/#1.0
ppk:305678.568/ISBN2890321345#1.0
ppk:305678.1000/ISBN2890321345-P235#2
```

一级基础 ODIN 可以采用短编码方式，结构式为：

```
ppk:[REG_ORDER_INDEX]/[DSS]
```

[REG_ORDER_INDEX] 为该 ODIN 记录在全部 ODIN 注册记录中以注册时间早晚

排序的阿拉伯数字索引值（从 0 开始）。

下面几例都是符合定义的一级基础 ODIN 的合法缩短编码：

```
ppk:1/
ppk:356/#1.0
ppk:356/ISBN2890321345#1.0
```

ODIN 的命名结构使每个数据资源在全网具有自主、安全的唯一索引标识。ODIN 不同于 URL，它是数据资源的索引名称，而与实际地址无关。实际上它是一种 URI 或 URN（Universal Resource Name，统一资源名称），是信息索引的数字标签和身份证。有了它，数据资源就具有了唯一性和可追踪性。

2. 多级扩展 ODIN 标识

以一级 ODIN 为基础，一级 ODIN 的拥有者可以利用自有的区块链来扩展自定义二级 ODIN，并将二级 ODIN 注册记录批量打包后形成的新区块的 HASH 关键字，写入上一级基础区块链，获得合法验证并确保唯一性。以此类推，可以形成更多级的 ODIN 标识。

多级 ODIN 的标准结构式为：

```
ppk:[PARENT_ODIN_PREFIX]/[SUB_BLOCK_SN].[SUB_TRANS_INDEX]/[DSS]
```

- [PARENT_ODIN_PREFIX] 为对应上级 ODIN 前缀。
- [SUB_BLOCK_SN] 和 [SUB_TRANS_INDEX] 为对应子级 ODIN 在上级自定义区块链上的登记记录所在区块和区块内记录位置的阿拉伯数字编号。
- 后缀 [DSS]（Data Suffix String）是由上级 ODIN 拥有者自选并自行给出的具体数据内容定位标识，需要自主确保具有唯一性，命名方案同上。

举例如下：

```
ppk:351474.430/21.35/
ppk:351474.430/21.35/ISBN2890321345#
ppk:351474.430/21.35/ISBN2890321345#1.0
ppk:305678.1000/23.678/235.32/ISBN2890321345-P218#
```

多级 ODIN 自定义结构式为：

```
ppk:[PARENT_ODIN_PREFIX]/[SUB_TRANS_ID]/[DSS]
```

[SUB_TRANS_ID] 为该 ODIN 记录在子级区块链上的唯一标识，由所属上级 ODIN 标识拥有者来定义，可以是流水编号，也可以是唯一取值的字符串，需自行保证能与标准结构式区分开，且不能包含“/”和“#”这两个字符。

举例如下：

```
ppk:351474.430/22/
ppk:1/22/ISBN2890321345
ppk:1/22/ISBN2890321345#2.1
ppk:1/china/books/
ppk:1/china/books/#
ppk:1/china/books/ISBN2890321345-P218#
```

10.2.5 ODIN 标识技术规范

类似 XCP（合约币）和 Mastercoin（万事达币）等数字加密货币的技术原理，ODIN 的实现是通过将特定消息数据按比特币协议规范进行编码后，作为比特币交易广播到比特币网络上，存入区块链。

每条 ODIN 信息包括以下特性：

- 一个比特币源地址（对应 ODIN 消息生成者）。
- 一个比特币目的地址（对应受 ODIN 消息指向的目标个体，当消息生成者与受消息指向的目标个体相同时，该地址为空）。
- 若干个 1-of-N 多重签名输出比特币地址公钥（由 ODIN 数据包编码生成，实际生成交易时从 ODIN 设置数据中按顺序每提取 31 个字节，并在该 31 个字节的前部加上第 1 个字节取值 3，第 2 个字节为后续有效数据长度，总共 33 个字节对应一个压缩格式的比特币公钥。如果不足 33 个字节的自动在尾部追加二进制 0 填满，直到正好达到 33 字节对应一个压缩公钥）。
- 比特币源地址中有一定数量的比特币余额。建议有 0.001BTC 以上，用于生成从源地址发送到目的地址的若干有效交易条目以嵌入 ODIN 数据包。注意：因为比特币 1-of-N 多重签名交易的特点，这些比特币金额不会发生实际支出，将在下一个 ODIN 消息中被回收循环利用。
- 以比特币计的消息成本固定费用（默认是 0.0001 BTC），将支付给收录这个交易数据块的比特币网络矿工。
- 一个比特币找零地址（与上述第一项的比特币源地址相同，用于按照比特币交易协议将输入交易的比特币金额在生成若干条满足嵌入 ODIN 数据包的交易条目后多出的金额回收到消息发送者账户）。

上述的特性第 3 项是技术实现的关键，ODIN 的数据会嵌入比特币交易的多重签名输出数据块中，因为是 1-of-N 输出，每个数据块的第 1 个公钥固定是发送者的，因而输出的币值可以赎回循环使用，第 2 ~ N 个公钥的地址空间用来存放编码的 ODIN 消息数据。关于比特币多重签名交易的详细说明请参考比特币协议规范。

注意：N 建议取值为 3，最大不超过 10。对 1 条 1-of-N 多重签名输出仍无法容纳的 ODIN 数据块，可依样扩展存入第 2 条、第 3 条等更多条多重签名输出记录中。

每个 ODIN 信息数据块的格式按字节顺序定义如下：

第 1 ~ 31 字节：前缀特征标识，31 个字节的 ASCII 字符串 "P2P is future! ppkpub.org->ppk:0"（不含两侧的双引号）。

第 32 字节：消息类型，1 个字节。

第 33 字节到消息结束为按消息类型区分的不同消息数据，详见 ODIN 技术规范中具体 ODIN 消息类型中的定义。

注意：为了便于识别，每个 ODIN 消息都以 31 个字节的指定字符串起始作为前缀特征标识，这个字符串非常长，因而不可能把 ODIN 的特定交易和其他的比特币交易搞混。

10.2.6　使用示例

通过 ODIN 标识解析并获取对应数据的方法很简单。

1. 通过一个 ODIN 检索网站或解析云服务

有网站搭建能力的个体都可以利用开源的 ODIN 检索库来设立一个 ODIN 检索网站或解析云服务。

比如，登录 http://ppkpub.org/odin/ 这个示例网站，在 Query ODIN 的提示框内输入已知 ODIN 号（例如 352084.951），单击 Go 按钮，ODIN 系统就会显示相应的 ODIN 注册信息，如图 10-11 所示。

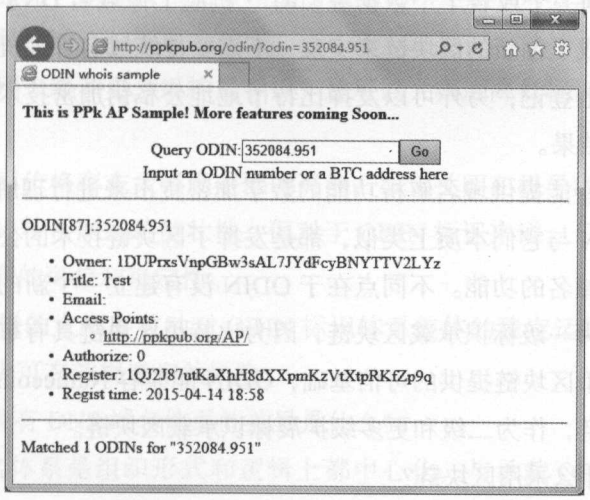


图 10-11　ODIN 注册信息

在此简单示例的基础上，可以进一步实现更为复杂的资源检索或云服务功能，读者可以留意 PPk ODIN 开源项目的具体进展。

2. APP 内置 ODIN 标识支持

APP 的开发者可以将开源的 ODIN 标识解析 SDK 直接包含到自己的程序内，这样就能直接解析和获取 ODIN 所对应数据资源的元数据描述和实际数据 URL，并根据元数据的定义来自定义最终展现。

10.2.7 开放资源

（HTML）ODIN（开放数据索引命名）技术规范如下：

```
http://ppkpub.org/ppk_odin_spec_cn.html
```

（GitHub）用 Java 语言编写的 ODIN 标识注册管理开源工具示例的源码如下：

```
https://github.com/ppkpub/javatool/
```

（GitHub）ODIN 标识解析应用开发工具包（SDK）：

```
https://github.com/ppkpub/sdk/
```

10.2.8 问题与思考

（1）如何类比已有案例以方便理解 ODIN 的功能？

ODIN 与域名 DNS 类比会比较容易理解些。

ODIN 会索引到一个或若干个数据源 URL，机制上像域名 DNS 解析，只是把传统的域名登记机构换成了创新的区块链来实现，不需要传统域名管理机构了，谁需要都可以自主地到区块链上登记，另外可以发挥比特币地址公私钥加密技术的特点，来提供更为安全可信的解析结果。

也可以类比一些能提供域名解析功能的数字加密货币来进行理解，比如 Namecoin、BTS 等，因为 ODIN 与它们本质上类似，都是发挥了区块链技术的公开可信、不可篡改的特点来提供类似域名的功能。不同点在于 ODIN 没有建立一个新的加密货币，而是明确比特币区块链作为一级标识承载区块链，因为比特币区块链具有事实上最强的稳定性和可靠性。以比特币区块链提供的可信基础，ODIN 能兼容 Namecoin、LTC 等其他区块链，甚至私有区块链，作为二级和更多级扩展标识承载区块链。

（2）ODIN 为什么采用区块链？

以比特币为代表的区块链的独特之处在于实现了历史上第一个形式上去中心化、逻

辑上却完美中心化的技术体系，简单理解就是：从彼此无关的不同节点读取到都是完全相同的唯一拷贝。以此为基础，才能定义出类似 ODIN 这样具有自主、安全关键特性的唯一资源标识。

（3）ODIN 采用比特币区块链承载一级标识是否能确保安全？

比特币作为第一个提出和实现区块链的加密货币，经过多年的运行已形成一个具有超强算力的分布式网络，其算力已经远超传统超级计算服务集群的总算力，而且还在持续增加，从而充分保证了其区块链的安全性和稳定性。

理论上，如果比特币网络的超大算力有超过一半被某个个体所控制，该个体就可以篡改近期的若干区块数据（即著名的 51% 攻击，参见第 9 章的内容），但其攻击难度随区块增长呈现指数级提高，超过 6 个区块确认后基本就不可能了。而且攻击者也只能篡改自己相关的交易信息（比如重复消费自己的比特币）或者不记录别人发出的交易，但不能凭空伪造别人比特币地址相关的交易，所以 ODIN 申请者在向比特币网络广播 ODIN 注册消息后，只需要等待 6 个区块就可以规避以上攻击风险以确认注册是否成功，即使在极小概率的情况下，比特币网络被攻击成功导致注册不成功，也只需重新发起注册，对于注册者来说除了耽误了一些操作时间之外没有损失。

所以采用比特币区块链作为 ODIN 的一级骨干区块链是安全可信的。

（4）比特币价格的大幅调整是否会对 ODIN 标识体系的稳定运行产生很大影响？

每条 ODIN 消息存储到比特币区块链的成本主要是支付给收录该交易的比特币“矿工”的费用，目前是 0.0001BTC，相当于约 6 美分或 0.4 元人民币，是非常少的。当比特币价格有大幅调整时，该项费用也可以适当调整，以达到相对合理和低的费用（调整客户端的参数配置即可）。

另外，通过使用二级和更多级扩展标识，还可以大幅度降低标识的注册和维护成本（可以接近 0 成本）。

如果未来比特币价格存在大幅度走低的可能，矿工关闭矿机导致算力大幅波动，在一定程度上会影响比特币网络的健壮性，但对于 ODIN 标识来说，只要等待 6 个区块的确认仍能保证相当高的可信和稳定性。

所以比特币价格的大幅度波动对 ODIN 标识体系整体的稳定运行只有很小的影响，而且通过适当的规则可有效规避相关风险。

（5）ODIN 与现有 DNS 域名体系的差异是什么？

现有 DNS 域名体系是组织形式和逻辑上都中心化，与承载 ODIN 的区块链相比，无法提供自主性，且 DNS 协议因为出现时期早，在安全性等多方面上也存在不足。但

因为其作为现有互联网的基础协议以稳定为重，很难做出大的改变。

ODIN 形式上和 DNS 域名有点像，但借助区块链的独特性使得其运行机制有本质的差别，它强调自主和安全，是“数据时代的自主域名”。

（6）ODIN 与 URI/URL 的差异在哪？

ODIN 不同于 URL（Uniform Resource Locator，统一资源定位符），它是数据资源的索引名称，而与实际地址无关，与 URL 的最大区别就是实现了对资源实体的永久性标识。实际上它是一种 URI 或 URN 是信息索引的数字标签和身份证。有了它，数据资源具有了自主、安全的唯一性和可追踪性。

（7）ODIN 与其他基于区块链的标识应用像 Namecoin、Onename 等案例的差异在哪？

ODIN 和以上这些标识应用本质上都利用了区块链技术来生成唯一可信标识，也都是完全开放开源的，但在技术方案、应用扩展上各有特点，差异对比如表 10-1 所示。

表 10-1 ODIN 与基于区块链的典型命名标识差异对比

	ODIN	Namecoin	Onename
基础区块链	明确以比特币区块链承载一级标识	采用域名币（Namecoin）自建的区块链	起初采用域名币（Namecoin）的区块链，后来因为域名币网络算力有限，存在被大矿池单方控制的问题，而切换到以比特币区块链为基础的 Blockstack 平台来承载
灵活扩展支持	支持扩展多级标识引入其他区块链技术（如公有链、联盟链、私链等）	无	支持整体迁移底层数据承载到其他区块链
命名方式	用区块记录位置作为名称标识，确保唯一性	抢注自定义字符串	抢注自定义字符串

10.3 本章小结

本章介绍了两个基于比特币的方案。介绍了闪电网络通过支付通道实现离线交易的原理，并且说明了如何在支付通道的基础上实现支付网络的功能，使得没有直接支付通道的比特币用户之间可进行离线交易。闪电网络的解决问题的思路被其他区块链系统借鉴，例如闪电网络已经被移植到以太坊上，用智能合约实现了雷电网络。另一个案例则是利用比特币网络实现的 DNS 方案——ODIN。ODIN 借助比特币强大算力的支撑，实现了去中心化的管理形式，确保了 ODIN 的 DNS 记录不可更改性。希望本章的案例能给读者带来更多的启发。

第11章

从架构变革看IT时代的演进

前面10章介绍了区块链的基础概念，对不同区块链架构进行了剖析，并对密码学、共识算法等关键技术做了详细的介绍；也提供了区块链中比特币和以太坊智能合约的开发指南；还介绍了区块链的应用案例，并讨论了区块链的常见问题。希望读者在读完这本书后，能在架构上对区块链有一个较为全面的认识，并掌握区块链开发的原理和基本技巧。本章将主要与读者分享笔者学习架构的心得，并就架构变革驱动IT时代发展，以及从互联网+到区块链+的架构演进，与读者分享一些个人的心得和观点。

11.1 架构心得

11.1.1 架构和技术的关系

笔者在IT行业从事架构师职业多年，养成了关注典型IT架构的职业习惯。如果把区块链涉及的技术拆开来看，其实无论是虚拟货币、密码学、共识算法，P2P通信等技术，大部分都有迹可循，先于比特币出现。区块链之所以变得这么重要，主要是由于它把这些技术进行有机的组合，形成一个去中心化的、自动化执行交易、自治管理的架构。因此，与其说区块链是技术的革新，不如说是架构的革新。

大部分人容易把社会的进步，时代的变革归结于某项技术的发明，而忽视了起关键作用的其实往往是架构的创新。回顾IT发展的历史，每次重大的变革，不无例外是由于新架构的出现产生了新的能力，带来了新的应用，应用推动了变革。架构和技术的关

系，有些像交响乐的指挥编排和各乐器演奏家的关系。就像一个杰出的指挥家指挥各种不同乐器演奏家协奏出气势恢宏的交响乐一样，一个好的架构，把提供不同技术功能的组件通过在空间、时间的排列组合，通过信息的沟通、协调，形成一个具备一定功能的完备计算系统。

11.1.2 关于计算的观察

IT 的实质是计算。在自然界，动物与生俱来掌握了计算的能力。这个从我们观察青蛙准确的捕捉蚊虫，候鸟在周期性迁移中对路径、时间、所需食物的准确把握，大型食肉动物对捕食对象以及捕食路径的选择取舍，可以很容易得出生物具备计算能力的结论。我们甚至可以从观察植物的某些特征，例如树根在地下的延展路径，树枝树叶的趋光性生长，甚至春华秋实的规律也可以推测出，植物也具备某种神秘的计算能力。因此，虽然没有很强的科学证据，但从计算的字面意义来说，可以在很大程度上假设在自然界中，生物具备某些计算能力。而非生物，也就是无机物，应该只是受物理和化学规律支配，并不具备原生的计算能力。

从某种意义上来说，动物的大脑类似电脑。但就从我们人脑的角度来看，纯粹的数字计算能力是非常有限的，两位数的乘法就使得大部分人皱眉。因此，从远古开始，人类就开始寻找帮助计算的工具。从结绳计数，到用小棍算筹计数，再发展到用算盘计算，其实是借助工具来进行计算。在这个计算过程中，计算的主体还是人，计算工具只是为了便于计数或记录中间结果，而没有成为参与计算的主体。到 17 世纪，法国数学家帕斯卡发明了齿轮传动的、能计算加减法的计算器，后来德国数学家莱布尼茨在帕斯卡的计算器基础上发明了能做加减乘除的计算器，这样，人类开始可以通过这些机械工具来进行计算，唯一需要的是通过手动来让机械计算器运转。再后来，机电式计算机、电子计算机相继出现，计算终于能由工具自动进行，代替人脑，唯一需要的是人的控制指令和预先输入的程序。

11.1.3 架构创新的神奇力量

从刚才所提到的计算机发展历史来看，每次技术革新，像机械技术、机电技术、电子管技术、半导体晶体管技术、集成电路技术、大规模集成电路技术，都给计算机的发展带来了推动力量，但真正具有划时代意义，起决定性作用的，还是计算架构的创新。这个从著名的图灵机模型就可见一斑。

英国科学家艾伦·图灵 1937 年发表著名的《论可计算数字，及其在判定性问题的应

用》一文（英文名 On Computable Numbers，with an Application to the Entscheidungsproblem.）。其中 Entscheidungsproblem 是德文“可判定性问题”的意思。“可判定性”意指对于一个判定问题，如果能够编出一个程序，以域中任意元素作为输入，执行该程序就能给出相应的个别问题的答案，就称该判定问题为可判定的。图灵在论文中提出了计算机抽象模型——图灵机的概念。图灵机由一个控制器、一条可无限伸延的带子和一个在带子上左右移动的读写头组成。这个在架构上如此简单的机器，理论上却可以计算任何直观可计算的函数。但同时，图灵也证明了不存在一个算法可以解决判定性问题。也就是说，有些计算问题是无解的。而所有能计算的算法都可以由一台图灵机来执行。图灵的理论证明了制造出能编程序来执行任何计算的通用计算机是可能的。

图灵机的意义在于，通过一个简单的架构，将原来只是生物具备的计算能力，赋予了无机物。这个是在人类历史上具有石破天惊影响和划时代意义的大事件。图灵机作为计算机的理论模型，奠定了现代计算机科学大厦的基础。

11.1.4　冯·诺依曼架构

1945 年，根据图灵机模型，匈牙利籍科学家冯·诺依曼提出了“存储程序”（Stored-program）的概念。“存储程序”指的是把程序指令和数据放在同一个存储器上，程序指令存储地址和数据存储地址指向同一个存储器的不同物理位置；采用单一的地址及数据总线，指令和数据宽度一样。处理器执行指令时，先从储存器中取出指令解码，再取操作数执行运算。程序计数器（PC）是 CPU 内部指示指令和数据的存储位置的寄存器。CPU 通过程序计数器提供的地址信息，对存储器进行寻址，找到所需要的指令或数据，然后对指令进行译码，最后执行指令规定的操作，程序按顺序执行。后来这种体系架构被称为“冯·诺依曼”体系架构，也叫“普林斯顿”体系架构，如图 11-1 所示。它由输入、输出设备，中央控制单元（CPU）、存储器以及连接 CPU 和存储的总线构成。

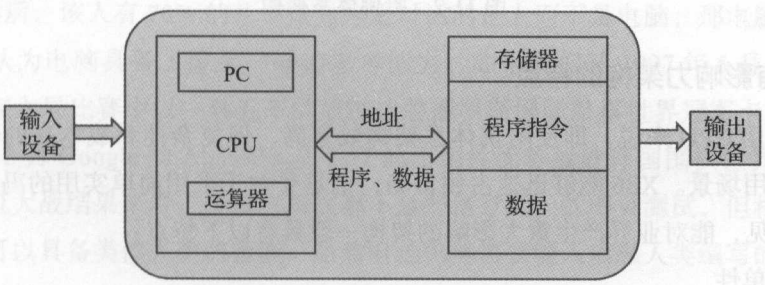


图 11-1　冯·诺依曼架构体系

由于程序指令和数据共用总线，因此冯·诺依曼体系架构的计算机并行能力差，数据处理速度慢，存在所谓的“冯·诺依曼瓶颈”。但该架构的特点是简单，不需要独立的程序存储和数据存储，大大减低了外设的复杂性。所以直到今天，这种架构仍是当前计算机的主流架构，绝大部分的计算机都属冯·诺依曼架构。典型的芯片像 Intel、ARM 的 ARM7 和 MIPS 等，都是支持冯·诺依曼架构的芯片。

11.1.5 哈佛体系架构

与冯·诺依曼架构不同的是哈佛体系架构。哈佛架构是一种将程序指令存储和数据存储分开的计算架构，如图 11-2 所示。中央处理器首先到程序指令存储器中读取程序指令内容，解码后得到数据地址，再到相应的数据存储器中读取数据，并进行下一步的操作（通常是执行）。程序指令存储和数据存储分开，可以使指令和数据有不同的数据宽度。哈佛架构的微处理器通常具有较高的执行效率。其程序指令和数据指令分开组织和存储，执行时可以预先读取下一条指令。目前使用哈佛结构的中央处理器和微控制器有很多，包括信号处理芯片（DSP）、摩托罗拉公司的 MC68 系列、Zilog 公司的 Z8 系列、ATMEL 公司的 AVR 系列和 ARM 公司的 ARM9、ARM10 和 ARM11。哈佛架构的优点是效率高，但架构复杂，对外围设备的连接和处理要求高，比较适用于外设少的应用场景，如嵌入式的单片机等。

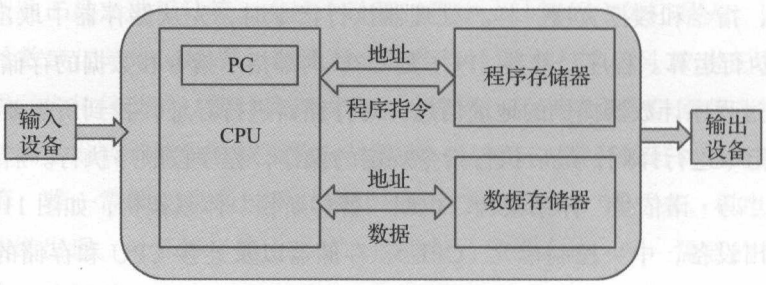


图 11-2 哈佛体系架构

11.1.6 有影响力架构的特点

对于通用 CPU 来说，即使哈佛体系架构效率高，但复杂性和成本高的特点决定了不适用于通用场景。X86 能够迅速占领市场，也是受益于采用简单实用的冯·诺依曼架构。由此可见，能对业界产生重大影响的架构一般具有以下特点：

（1）简单性

无论是图灵机模型还是冯·诺依曼架构，给人的第一印象是其简单性，也就是说

构成架构的组件比较少，组件间的协同也比较简单，依托此架构做的产品也比较简单易用。

（2）完备性

构成架构的组件恰好不多不少，缺一不可，也不需要增加任何另外的元素就可以完全解决目标领域的所有问题。

（3）通用性

架构都具有一定的适用场景，太专用的场景不太可能产生影响力，因此通用性是形成主流架构的必要条件。

（4）权衡性

能流行的架构都会在效率和成本之间找到最佳的平衡点。只强调技术的先进性而忽视成本往往是很多产品失掉市场的原因。

（5）兼容性

好的架构一定是开放、兼容的架构。其实冯·诺依曼架构和哈佛架构也不是绝对对立的架构。现代的冯·诺依曼体系的芯片也吸收了很多哈佛架构的特点，比如 CPU 中的指令缓存等，都有哈佛架构的影子。

（6）扩展性

判断一个架构能否流行并可持续发展，最重要的指标是看其扩展性。好的架构，其扩展是通过简单的复制、排列和组合，将把最基础的系统做线性的扩展，使之功能和性能产生线性的增长。

11.1.7　从非生物计算到非生物智能

1950 年，图灵发表题为《计算机器与智能》（英文名《 Computing Machinery and Intelligence 》）一文，第一次提出了衡量机器是否会思考的标准——“图灵测试”。简单来说，图灵认为，如果把一个人和电脑隔开，让他们通过屏幕打字来对话，如果在持续 5 分钟的对话后，该人有 70% 的几率认为与之对话的是人而不是电脑，那电脑就通过了测试，可以认为电脑具备了像人一样的思考能力。如果我们从 1997 年 5 月 11 日 IBM 的深蓝电脑在六局比赛中以一胜五平的战绩战胜俄罗斯国际象棋世界冠军卡斯帕罗夫，还有 2016 年 3 月 Google 的 AlphaGo 以 4 : 1 的压倒性优势战胜韩国围棋世界冠军李世石这两次人机大战结果来看，虽然这两次都不是严格意义上的图灵测试，但相信很少人会质疑电脑可以具备类似人类的智能。虽然电脑仍然需要输入依靠人类编写的程序，另外很多人类做起来比较简单的任务，例如模式识别、情感分析等，对电脑来说还很难，

但其深度学习能力以及决策能力已经不容置疑。

可以说图灵在60多年前就预测到，作为人造无机物的电脑，通过在简单的图灵机架构的基础上的排列、组合，可以在某些方面逐渐演化成和人类智能匹敌的机器。到今天，大数据、人工智能（AI）以及机器人的发展，已经使得我们正式步入了一个无机物智能的时代。凯文·凯利在《失控》里的一个重要思想就是，人造无机物表现得越来越像生命体。一个名词“硅基智能”也应运而生，相对于作为生物的“碳基智能”而言，“硅基智能”可能更强大。因为从进化速度上，显然“碳基智能”没法和以摩尔定律速度发展的“硅基智能”相提并论。因此很多人认为，无机物智能超过人类智能会是一个必然很快发生的事情。所以，科学家霍金，企业家、特斯拉董事长马斯克，以及比尔·盖茨都对人工智能的发展态势表示担忧就不足为奇了。在2015年7月于阿根廷举办的2015年国际人工智能联合会议（IJCAI）上，超过千位科学家签署公开信，敦促联合国禁止开发自主性武器（autonomous weapons）。霍金、马斯克、沃兹尼亚奇（苹果创始人之一）等都进行了签署，第一次对人工智能所带来的威胁发出警告。

上面的一些心得其实是在多年来从事架构师工作中受到的一些启发。下面想谈谈对从互联网+向区块链+的发展趋势的一些观察和看法。

11.2 架构创新——IT发展源源不断的动力

前面提到，图灵机模型奠定了现代计算机的理论基础，而冯·诺依曼架构主导了60多年来计算机的发展。在这60年中，宏观层面的IT架构发生了几次大的变化。如果从一个构成IT系统的最底层的视角来看，这些大的变化可以归结为，在新的场景中为突破底层冯·诺依曼瓶颈而进行的上层架构调整。这个怎么理解呢？从单个计算机来看，冯·诺依曼架构在效率上的局限性是很明显的，主要表现在指令串行执行、程序和数据共享总线、计算和数据分离方面。而在IT层面碰到的问题，包括性能和扩展性问题，需要通过把底层的冯·诺依曼架构的电脑采用特定的协议按一定架构进行组合，以达到在整体上满足业务的功能和非功能方面的需求。

如果我们回顾IT架构的发展历程，我们可以将其大致分为传统集中式的大中型机时代、以PC为主的客户端/服务端（Client/Server）时代、互联网时代、云计算、大数据时代，到现在是互联网技术和传统行业深度融合的互联网+时代。下面我们简单介绍各个时代的情况。

11.2.1　大中型机时代

在集中式的大中型机时代强调的是单机的能力、垂直扩展的能力。为了更好地提升资源利用率，各个层面的虚拟化技术，包括 CPU 分时系统、内存虚拟化、计算虚拟化等创新都在大型机时代出现。

在 20 世纪 60 年代，美国电脑科学家，也是首先提出人工智能（Artificial Intelligence）概念的约翰·麦卡锡（John McCarthy）就提出了公用计算（Utility Computing）的概念，预言将来计算也可以像水和电那样作为公用服务提供。这个就是我们今天耳熟能详的云计算理念。

大型机的集中化管理以及从大型机产生的虚拟化技术，孕育了云计算的最早概念。但随着信息化的普及，大型机无论从成本、开放性和扩展性方面都不能满足日益多元化、普及化的信息处理的需求。

11.2.2　开放时代的到来

20 世纪 70 年代，随着 UNIX 操作系统和 C 语言的出现，开放、互联的理念开始萌芽，给 IT 架构的发展带来一线曙光。另外从大中型机和终端之间通信，到大中型机之间的通信的需求催生了网络技术的发展。在这个过程中，以太网技术、TCP/IP 技术相继诞生。特别是在 1981 年，国际标准化组织（ISO）提出了开放体系互联架构的基本参考模型（OSI/RM），这个就是著名的 OSI 开放网络七层架构，奠定了在不同系统的不同应用相互进行透明通信的基础。

11.2.3　客户端 / 服务端（CS）分布式时代

到了 20 世纪 80 年代，个人对计算的需求催生了 PC 机时代的到来。随着 PC 的普及，局部联网的需求也越来越大。局域网的发展催生了客户端 / 服务端（Client/Server，CS）分布式架构。客户端 / 服务端架构实际上是把一部分原先在集中式主机中的工作负载分配到客户端上运行，减轻了主机的负担，提升了整个系统的并行处理能力。这个相当于在不改变每个电脑的冯·诺依曼架构的前提下，用在多个冯·诺依曼架构的小型电脑上并行运算，来取代一个在集中式的大型机里的串行计算。

分布式架构的一个重要特点是把计算、存储分布在网络中的多个节点上，通过软件来控制任务的分发和执行调度，这样任务能同时在多个节点上并行执行，同时给上层的应用提供一个访问远程节点如同访问本地系统一样的接口环境。早期的远程程序调用（RPC）、分布式计算环境（DCE）、普通对象请求代理（CORBA）等都是分布式框架。这

些分布式框架强调的是计算的分布。分布的范围也是在企业的内网。

第一个分布式存储是是由 DEC 公司在 20 世纪 70 年代开发的分布式文件系统 File Access Listener（FAL）。SUN 于 1985 年发布第一个广泛使用的分布式文件系统 NFS。其后像 AFS、KFS、DFS、IBM 的 GPFS、SUN 开发的 Lustre 等如雨后春笋般出现。

分布式架构改变不但节省了成本，还提高了效率，使得计算更靠近客户，而不是像过去那样，客户必须到与大中型机相连的终端上才能提交计算任务。

11.2.4 互联网时代

到了 20 世纪 90 年代，互联网开始普及。互联网突破了局域网的限制，使得计算可以跨越时空的限制。CS 架构逐渐向 BS 架构（Browser-Server，浏览器 - 服务端）转型。BS 架构将客户端统一到浏览器，使得应用在任何平台环境下都可以运行。这一时期，IT 架构最关注的是开放性和可移植性，因为 IT 系统的资源非常宝贵，希望应用开发一次，可在所有 IT 系统上运行。在这个理念下，催生了 Java 这个高移植性的高级编程语言。

进入 21 世纪，随着应用越来越多，IT 架构开始强调组件的专业化和分工协作。一个著名的架构原则叫“ Separation of Concerns”（SOC），意思是对不同的问题要用不同的组件分开处理。这期间一个名叫“Model-View-Controller”（MVC）的三级架构开始流行，该架构就很好地体现了 SOC 原则。在 MVC 中，Model 负责数据实体操作，View 负责呈现，Controller 负责控制逻辑。这个三级架构可以用在各个层面，小到简单的网站应用，大到企业 CRM 或 ERP，MVC 三级架构都得到广泛应用。

而互联网中最著名的 HTTP 协议的主要起草人 Roy Fielding 在他 2000 年完成的博士论文中提出了 REST（Representational State Transfer）的架构理念，该架构理念奠定了互联网的架构基础。意思是“表示性状态传输”，单从字面上不好理解它的意思。实质上它是把世界上所有资源都用一个唯一的标志表示（Universal Resource Identifier，URI），对客户端每次请求，服务端都回复一个资源的表示性状态，而不是资源本身。

举个例子，用户在网页上浏览一条新闻，该请求会将指向新闻资源的 URI 发到服务端。服务端将根据 URI 找到该条新闻资源，并将其“表示性状态”，也就是用 HTML 或 XML 格式编写的新闻内容反馈给客户端；而新闻资源本身还是留在服务端。

在 REST 架构下，客户端 / 服务端的多个请求之间没有上下文的关联，上例中新闻资源的内在表示状态和回复给客户的表示状态就没有必然联系，如服务端的新闻资源可能是以数据库的形式存放的，接下来用户可能发一个观看视频的请求，这个请求和上面的请求完全无关，没有上下文的关联关系。服务端在接到请求后，将视频资源的“表示

性状态”，也就是视频码流发给客户端。用户也可以通过发指令来修改资源的“表示性”状态或删除资源。没有“上下文”关系，也就意味着请求是无状态的，这样也就意味着可以不用担心状态的管理，而方便地扩展支撑互联网的基础设施，比如增加 Web 服务器来增强处理请求的能力等。

REST 架构是超媒体（Hypermedia）浏览的最佳选择。超媒体就是用超链接（Hyperlink）的方式，将各种不同空间的文本或多媒体信息组织在一起的网状媒体，互联网的网页就是一个超媒体。REST 架构在扩展性上具有无与伦比的优势，最后成为互联网的主流架构。REST 架构能够成为主流，最重要的归结于它在架构上的简单性、兼容性和扩展性。当时以 IBM 和微软为首主推的基于 SOAP 的 Web 服务架构就因为过于复杂，而逐渐被 REST 边缘化。

分布式系统要比集中式系统更复杂，因为分布式系统要解决的问题是一致性（Consistency）、可用性（Availability）和分区容错性（Partition Tolerance）。一致性指的是在同一时刻，在每个节点都能读到最新写入的数据。可用性指的是一个运行的节点在合理的时间内总能响应请求，不会发生错误或超时。分区容错性指的是当网络分裂发生时，系统仍能继续工作。这些问题在集中式系统中都不是大问题。但在分布式系统，特别是在大规模的互联网分布式系统中却成为最大的挑战。

2000 年，Eric Brewer 教授提出了一个猜想，就是一致性、可用性和分区容错性三者无法在分布式系统中被同时满足，并且最多只能满足其中两个。而后这个猜想被证明，上升为大名鼎鼎的帽子理论（CAP Theorem）。CAP 理论给分布式系统设计者的启发是：任何一个分布式系统的设计都要根据应用场景的不同而对一致性、可用性和分区容错性做取舍权衡，三者取其二，不能什么都想要，什么都不想舍。

传统企业级 IT 应用主要处理的是交易型数据，比如账务、库存数据，因此一致性是根本要求。大部分企业应用需要满足 ACID 要求。A 代表 Atomic，原子性，也就是一个交易要么全完成，要么全不完成，不能处在一个中间状态；C 代表 Consistency，一致性，指的是在交易开始前或结束后，关联数据必须保持一致；I 代表 Isolation，隔离性，也就是不同交易必须相互独立，完全隔离；D 代表 Durability，持久性，指的是交易后的数据能持久保存。

到了互联网时代，互联网应用主要是处理交互型数据，像分享的网页、图片等。交互型的数据比交易型的数据的体量大很多，这些数据对一致性没有很高的要求。但对计算的处理能力的要求比传统计算的要求要大得多。互联网环境和企业内网环境不一样，网络出现问题是常态而不是异常现象，因此互联网应用对可用性和分区容错性有很高要

求，大部分互联网应用需要满足 BASE 要求。BA 代表 Basic Available，基本可用，也就是即使某个节点发生故障或网络断开发生，系统应该仍能正常工作而不报错；S 表示 Soft State，软状态，也就是暂时状态，指的是在某一时刻的某些节点上，允许不一致的暂时状态出现；E 代表 Eventual consistent，最终一致，也就是系统最后通过补偿机制或者其他纠错机制使得最终状态保持一致。

相比 ACID 应用，BASE 的应用更具良好的扩展性，更适合于在互联网上运行的分布式系统。在互联网上如何做 CAP 三者的权衡？如何支持 BASE 的应用？如何满足处理不断增长的海量数据的计算能力的需求？

11.2.5 云计算、大数据时代

1. 谷歌的架构变革

从 2003 年到 2004 年，谷歌（Google）陆续发表了关于 GFS、MapReduce 和 BigTable 的 3 篇论文，基本上公开了谷歌内部用于处理搜索海量数据的平台架构。GFS 是大规模的分布式文件系统，MapReduce 是一个并行处理框架下的编程模式，BigTable 是建立在 GFS 基础上一个按键值方式组织的非关系型数据库。由于当时的技术、产品和平台无法满足谷歌快速增长的业务发展，谷歌根据搜索业务的特点，大胆创新，打破了传统分布式文件系统的条条框框，开发了一个支持大规模扩展性的容错分布式文件系统，并在其基础上构建了并行计算平台和分布式数据库，使得谷歌的搜索平台能处理前所未有并不断爆炸性增长的海量数据。

特别是 MapReduce 的并行处理编程框架，通过软件对数据进行自动的切分，并把任务分配到不同节点上，实现自动调度、均衡工作负载，同时自动监控，自动修复错误，管理节点间通信。传统的并行处理应用，需要开发者掌握 MPI 编程等技能，一般只是限于高性能计算领域。而 MapReduce 框架简化了并行处理系统的编程，大大降低了开发者开发并行处理系统的门槛。可以说，MapReduce 框架，使得先天缺少并行计算能力的冯·诺依曼架构的电脑，通过集群的并行处理，仍然能够在互联网时代焕发青春。

谷歌的 3 篇论文奠定了互联网大规模分布式系统的架构基础，掀开了大数据时代的帷幕。谷歌的贡献主要是基于其自身的业务需求，在对比传统分布式架构优劣势的基础上，提出了一套全新的分布式存储、分布式并行计算和分布式数据库的架构。但其特点还是在集中化模式管理下的可扩展分布式系统。

2. 亚马逊的架构变革

谷歌是首先提出云计算概念的公司，而另一个首创云计算业务模式的亚马逊也不甘

落后，于 2007 年发表了 Dynamo 分布式数据库论文。与谷歌相同的是，亚马逊也是根据自身的业务特点来做创新，都将系统出错作为常态处理；而与谷歌不同的是，亚马逊采用了一个无中心、完全分布式的架构。

亚马逊的 Dynamo 论文公开了分布式键值数据库 Dynamo 的设计和实施细节。Dynamo 的设计主要是针对大规模电商的应用场景，例如购物车，需要提供“Always on”（总是在线），任何时候用户都能修改，也就是高可用的客户体验。其设计目标是把可用性提到第一位，在某些场合牺牲一致性。Dynamo 论文很明确的提出“Eventual Consistency”（最终一致性）的概念。其设计理念参考 Peer-to-Peer 架构，整个分布式系统采用无中心架构。Dynamo 综合了一些著名的技术来实现可伸缩性和可用性：数据划分（Data partitioned）和使用一致性哈希的复制（replicated），并通过对象版本（object versioning）提供一致性。在更新时，副本之间的一致性是由仲裁（quorum）中心化的副本同步协议来维持的。Dynamo 中共涉及 3 个重要的参数，其中 N 代表数据的副本数，W 代表一次写操作的最小必须写成功节点数；R 代表一次读操作的最小读成功节点数。要求 W+R>N，读数据时，只要有除了 Coordinator 之外的 R–1 个节点返回了数据，就算是读成功（此时可能返回多个版本的数据）。同理，写数据时，只要有除 Coordinator 之外的 W–1 个节点写入成功，就算数据写入成功。Dynamo 采用了基于 gossip 协议分布式故障检测及成员（membership）协议。Dynamo 只需要很少的人工管理，存储节点可以添加和删除，而不需要任何手动划分或重新分配（redistribution）。Dynamo 很早就成为 Amazon 电子商务平台的核心服务的底层存储技术，它能够有效地扩展到极端高峰负载，在繁忙的假日购物季节也没有任何的停机时间。

Dynamo 和 BigTable 都属于非关系型数据库，也就是常说的 NoSQL 数据库。但两者设计理念有很大的不同。Dynamo 是完全无中心的设计，其假设是在内部信任网络部署，没有安全的措施。而 BigTable 是集中式管理，利用权限控制来提供安全措施。Dynamo 的数据模型是键值模型，而 BigTable 是多维排序图。Dynamo 采用一致性哈希来实现分布式元数据管理，而 BigTable 采用集中式的元数据管理。两者的适应场景也各不相同。Dynamo 主要针对电商购物车应用，对可用性要求高，一致性要求不高，在 CAP 上强调对 A（可用性）和 P（分区容错性）的要求，是一个典型的 AP 数据库。而 BigTable 对一致性和可扩展性的要求比较高，比较适合处理结构化的数据，是一个典型的 CP 数据库。

3. 云计算架构的特点

云计算双雄谷歌和亚马逊开启了云计算、大数据时代。但云计算、大数据的概念却

很快沦为厂商炒作的概念，使得很多客户感到困惑。

美国的标准和技术组织（NIST）给了一个比较客观、经得起时间检验的定义，可以澄清很多误解。NIST 给出的定义是："云计算是一个提供泛在、方便、按需，并通过网络访问一个共享的可配置的计算资源池（包括网络、服务器、存储、应用和服务）的模式，该模式下资源能够迅速地被创建或释放，而不需太多的管理开销或服务提供商的人工干预。"

NIST 还进一步总结出一个"三四五"要点来进一步阐释云计算概念："三"是指云计算的 3 种服务模式（基础设施即服务 IaaS、平台即服务 PaaS、软件即服务 SaaS）；"四"是指云计算的 4 种部署模式（私有云、公有云、混合云、社区云）；五"是指云计算的 5 个特点（按需自服务、宽带访问、资源池、快速弹性扩展、计量服务）。

前面我们在架构的特点中提到，架构不单单是技术的简单组合，其中很重要的是要权衡效率和成本，也就是说，在架构中的诸多考虑因素中，一个很重要的考虑因素是经济的因素。缺少合理经济模型的架构，是很难成为主流架构的，从冯·诺依曼架构和哈佛架构的对比可以看出这一点，从云计算的架构特征来看也是一样。

NIST 的云计算定义还是太正式，太文绉绉地不好理解。用大白话来解释，云计算其实就是用容错、并行调度软件来把大规模的廉价的标准工业服务器组成资源池，把资源池的 IT 能力（具体包括计算能力、网络能力、存储能力、应用能力）转化成服务，以弹性按需的方式提供出去。这里面有几个值得关注的地方：

1）云计算是在分布式架构的基础上融入集约化管理的能力，具有集中化的架构特点。云计算时代，大部分的业务逻辑、数据处理都集中在运行于大型数据中心的云上，移动端主要是做展现。互联网时代的浏览器 / 服务端（BS）架构也逐渐向客户端 / 云端（CC）架构转型。

2）云计算架构是成本和效率的权衡。通过使用廉价的标准工业服务器，而不是昂贵的品牌设备来降低硬件成本，同时利用软件的容错来弥补廉价硬件质量的问题，并采用并行、虚拟化技术来提升资源的使用效率。

云计算是一种方便的提供计算、网络、资源以及在此之上构建的 IT 能力的一种服务模式。在该模式下会形成一个云计算所特有的架构特点。这些架构特点就是面向服务（Service Oriented Architecture）、资源池化（Resource Pooling）、软件定义（Software Defined）、标准化廉价硬件（Commodity Hardware）、计量服务（Measured Service）、水平扩展及弹性（Scale-out & Elastic）。其中软件定义和标准化廉价硬件对传统 IT 的冲击最大。软件定义网络、软件定义存储、软件定义安全的一个重要特点就是将控制层面与

数据层面分离，将控制软件与硬件解耦，这样可以大幅度提高系统的开放性、扩展性和灵活性，也使得管理更为方便。另外一个从经济层面考虑的因素就是能大幅度降低成本，让传统 IT 网络、存储和安全厂商失去专用硬件的保护壁垒。而以标准化廉价硬件为目标的 OCP（Open Compute Platform）以及天蝎计划则把话语权从传统厂商方面转到最终用户方面。不难想象，Cisco、EMC 等以生产品牌产品为主业的传统厂商将会面临来自“白牌机”厂商的巨大挑战。

4. 大数据产生的根源

BigData 这个概念是在麦肯锡咨询公司在其 2011 年 5 月发表的报告《Big data：The next frontier for innovation，competition，and productivity》中首次提出。在其报告中给出的大数据定义是：大数据指的是大小超出常规的数据库工具获取、存储、管理和分析能力的数据集。

大数据概念一经提出，迅速占领媒体的封面，出现了各行业言必称大数据的局面。很多传统的 BI、数据仓库方案也被重新包装，以大数据面目出现，甚至很多与大数据无关的项目，也被包装成大数据。大数据一时间成了一个包装标签。

国际数据公司（IDC）从大数据的 4 个特征来定义它，即海量的数据规模（Volume）、快速的数据流转和动态的数据体系（Velocity）、多样的数据类型（Variety）、巨大的数据价值（Value）。亚马逊的大数据科学家 John Rauser 则给出了一个简单的定义：大数据是任何超过一台计算机处理能力的数据量。

其实大数据的定义可能不是很重要，毕竟不同的角度观察可以有不同的定义。但有一个现象确实是不争的事实，数据量的指数型爆炸性增长给人类带来了前所未有的挑战。这个挑战表现在计算能力上，表现在支持计算的能力的能源消耗上。因为，计算后面需要有能源的支持。理论上来说，数据的增长永远会在越来越短的时间里倍增，但能源却没有办法跟随。最后结果只能是产生的数据被废弃。根据思科的预测，全球数据中心每年的 IP 流量会在 2019 年达到 10.4ZB，平均每月 863EB，几乎 3 倍于 2014 年的（2014 年是 3.4ZB，平均每月 287EB）。这种增长速度将直接推动 IT 架构的创新和变革。

那为什么大数据的挑战会突然在 21 世纪初期出现呢？其实，冰冻三尺，非一日之寒，大数据现象是持续几十年的人类社会信息化、数字化的结果。回顾二三十年前，大部分的通信信号都是模拟信号。但数字化革命后，大部分的信号都从模拟信号转化成数字信号。IT 的发展更加速了数字化的进程。早期的电脑只是把业务和管理信息化、数字化；互联网和移动互联网使得人与人的交流数字化；到了物联网时代，物与物的交互所产生的数据量会变得比前者更大。

大数据的出现，要求更大的计算处理能力，从而推动了IT架构的发展。Google的3篇论文实际上就是在这种背景下产生。

目前大部分人看到的是大数据带来的机遇，希望通过大数据分析，能更准确地掌握客户需求，能够更好地把握市场变化的脉搏，能更快地通过大数据的辅助决策来响应业务的变化。如果说，大数据的处理能力需要云计算来支撑，那么，大数据的分析，就要与行业知识相结合，建立相应的行业大数据分析模型。而与行业进行深度结合，也催生了互联网+时代的到来。

11.2.6 互联网+时代

互联网+时代实际上云计算和大数据时代的一个延伸，是云计算和大数据技术和行业深度融合的一个阶段，其实质是企业数字化转型。

1. 互联网+——企业数字化转型

IDC认为，IT行业正在进入以云计算、移动互联、大数据和社交媒体为代表的第三平台时期。IBM也提出向CAMSS（Cloud，Analytics，Mobile，Social，Security）领域全面转型。在国内，“互联网+”成了2015年一个最凝聚共识的词。上至国家领导人，下到老百姓，都在热议“互联网+”。“互联网+”的一个重要特点是实现传统企业的数字化转型，而云计算是承载企业数字化的架构和平台基础。

对于“互联网+”的内涵和外延有很多争论。比较趋同的看法是：“互联网+”是互联网时代从消费互联网向产业互联网过渡的一个重要阶段，其主要特点是传统行业采用“互联网思维”来创新业务模式，利用互联网和大数据技术，通过线下、线上的紧密结合，为客户提供更好价值的服务和产品。如此一来，也产生了各种互联网技术与行业结合的模式，例如：互联网+金融、互联网+制造、互联网+教育、互联网+金融、互联网+交通、互联网+能源等。

2. 互联网+金融——Fintech

其中互联网+金融成为最引人关注的领域。在国外，与互联网+金融对应的概念叫Fintech（中文意思是“金融科技”）。Fintech最早源于用于大型金融企业的后台的IT技术，包括账务系统、交易平台、支付、结算、清算等技术。进入互联网时代，Fintech的概念外延延伸到覆盖支撑金融行业业务创新的IT技术，包括P2P借贷、众筹、移动支付、虚拟货币、客户行为大数据分析等。典型的代表是P2P借贷的Prosper和LendingClub，移动支付的谷歌钱包、苹果钱包、阿里的支付宝和腾讯的微信支付、资金整体管理平台Mint、智能理财顾问（Robo-advisor）LearnVest，以及比特币Bitcoin等。

3. 区块链——Fintech 的天之骄子

前面我们谈到，一部 IT 架构发展的历史，经历了从大型机集中式，到 CS 分布式，再到云计算集中式的发展。分久必合，合久必分，历史不是简单的重复，而是以螺旋上升的轨迹发展。我们看到，谷歌的 3 篇论文都是关于以集中式的架构来管理分布式的计算。这样的好处是统一了元数据管理和调度，同时保证了一致性。而亚马逊的 Dynamo 架构，则有明显的去中心化的特点。中心化架构的一个很大的问题就是管理节点的性能瓶颈，容易成为攻击目标。还有一个最重要的问题就是在一个大的分布环境里建立、维护中心节点的信任所需要的成本非常大。

中心化的架构还有一个重要问题就是，如果管理中心节点的人的发生主观错误，或有诚信问题，或者受第三方影响而失掉独立性，将会对整个网络带来灾难性的影响。比特币的发明人中本聪于 2009 年 1 月 3 日在挖出的第一个创世纪比特币区块中留言：“ Chancellor on brink of second bailout for banks”。这句话是当天在英国泰晤士报登的封面头条新闻，中文意思是“财长处于第二轮银行紧急救助的悬崖边缘”。当时所处的背景正是席卷全球的金融危机的愈演愈烈的时期。据普林斯顿出版的《比特币和密码学技术》一书的作者分析，中本聪是出于对中心化的银行体系滥发货币、不加节制的扩张信用不满，而开发了一个完全去中心化的虚拟货币系统。中本聪一开始就开放比特币系统源码，比特币系统也不受任何人控制，比特币系统的总货币发行量也设计成固定的 2100 万比特币，按一定的规则逐渐发行。因此，比特币像黄金那样，具一定的稀缺性，是一个不会通胀的虚拟货币。

比特币自 2009 年上线以来，已经不间断地正常运行了 7 年多。比特币的试验证明，完全无中心化的分布式架构可以在陌生环境下通过合适的经济模型（挖矿激励）和共识算法形成信任。这就规避了中心化分布式架构在中心节点的致命弱点。同时，比特币底层的区块链架构也解决了一个互联网无法解决的问题，就是高昂的信任建立和维护成本。另外区块链通过密码学的签名、哈希算法解决了在互联网上难以解决的防伪问题。还有一个不太引人注意的独特地方是，在区块链上的计算需要用“燃料”（Gas）或交易费支撑，也就是说，计算与支撑计算的成本绑定。这和传统 IT 架构有很大的不同。在传统 IT 架构中，没有金融的元素。这样做的隐患是可以通过计算来攻击计算，这也是目前在互联网上无法杜绝的“拒绝服务”（DDoS）攻击的原因。而在区块链上，DDoS 攻击的可能性大为减少，因为发动 DDoS 攻击需要动用很大的虚拟货币储备。这个无论从成本上还是攻击源的掩饰上都会给黑客带来很大的不利影响。因此，区块链是天然的和金融紧密结合的 IT 架构。

更重要的是，结合脚本引擎、密码学和虚拟货币机制，区块链上可以实现支付、自动结算和清算。因此，区块链也被 Northwest Passage Ventures 公司的 CEO Alex Tapscott 称为“价值互联网”。

因此，区块链的意义是不言而喻的。特别是对金融行业来说，信用风险是传统金融业中一个挥之不去的梦魇。但区块链却带来了解决信用风险的近乎完美的解决方案。因此，区块链技术被认为是下一代互联网颠覆性技术也就不足为怪了。华尔街日报甚至宣称，区块链是最近 500 年以来在金融领域最重要的突破。因此，区块链可以说是 Fintech 领域中当之无愧的天之骄子。

11.2.7 区块链 + 时代

互联网 + 时代是互联网技术和行业业务的深度融合，但互联网在信任的建立、维护以及安全上存在致命的先天缺陷。未来互联网 + 必须与区块链 + 相结合，才能弥补这个缺陷。区块链架构的独特之处在于：

- ❑ 去中心化
- ❑ 公正性和透明性
- ❑ 防伪、防篡改
- ❑ 准匿名性
- ❑ 全网共识机制
- ❑ 交易可追溯
- ❑ 状态全网记录
- ❑ 安全性
- ❑ 合约自动执行
- ❑ 低成本及高效率

根据这些特点，区块链可以和很多行业结合，从当前的互联网 + 向区块链 + 发展，使得业务交易更安全，交易成本更低，交易效率更高。

1. 区块链 + 金融

区块链在金融行业无疑会得到广泛的应用。在支付、结算、清算领域，区块链可以成为“杀手级”的应用。例如在多方参与的跨地域、跨网络支付场景中，Ripple 支付就是一个很好的案例；在多方参与的结算、清算场景，R3 联盟也在利用区块链技术构建银行间的联盟链。同时在多方参与的虚拟货币发行、流通、交易、股权（私募、公募）、债券以及金融衍生品（包括期货、期权、次贷、票据）的交易（NASDAQ Linq 平台案例），

以及在众筹、P2P 小额信贷、小额捐赠、抵押、信贷等方面，区块链也可以提供公正、透明、信用托管的平台。在保险方面，区块链也可以应用于互助保险、定损、理赔等业务场景。

2. 区块链 + 政府

区块链防伪、防篡改的特性能够广泛用于政府主管的产权、物权、使用权、知识产权和各类权益的登记方面，包括公共记录，如地契、房地产权证、车辆登记证、营业许可证、专利、商标、版权、软件许可、游戏许可、数字媒体（音乐、电影、照片、电子书）许可、公司产权关系变更记录、监管记录、审计记录、犯罪记录、电子护照、出生死亡证、选民登记、选举记录、安全记录、法院记录、法医证据、持枪证、建筑许可证、私人记录、合同、签名、遗嘱、信托、契约（附条件）、仲裁、证书、学位、成绩、账号等方面的记录登记。

3. 区块链 + 医疗

区块链在医疗行业中可以应用于诊断记录、医疗记录、体检记录、病人病历、染色体、基因序列的登记，也可以用在医生预约、诊所挂号等应用场景，以建立公平、公正透明的机制。另外在药品、医疗器械及配件来源追踪、审计方面也有比较好的应用场景。

4. 区块链 + 物联网

利用区块链的智能合约，可以通过接口和物理世界的钥匙、酒店门卡、车钥匙、公共储物柜钥匙做程序的对接，可以达到区块链上一手交钱、物理世界一手交货的原子交易的效果。区块链在物联网的应用非常广泛，特别是在智能设备的自主管理，以及智能设备之间的互联、协调方面有着非常大的优势。

5. 区块链 + 商业

区块链在商业上的应用也非常广泛。凡是涉及交易、支付、积分等的场景都是比较适合区块链的应用场景。这里包括用区块链技术来实现打折券、抵用券、付款凭单、发票、预订、彩票、球票、电影票等业务流程的去中心化管理，以达到降低成本、提升效率的目的。

6. 区块链 + 能源

区块链在能源行业的应用前景广阔。采用区块链技术，可提供公正、透明的能源交易多边市场和碳交易市场，以达到降低对手信用风险，同时减少支付和结算成本、提高效率的目的。另外在缴费领域、分布式发电，特别是新能源微电网中发电家庭、用电家庭和电网间的电交易，区块链都是非常理想的技术。区块链也可以用来记录发电、配

电、输电、调度、用电、售电记录，提供公正、可追溯、透明的审计、监管记录。更重要的是，区块链在未来智能电网、能源互联网中会扮演更重要的角色，理论上可以通过区块链智能合约实现发、输、变、配、用电的同步调控。

区块链在别的行业，像电信、教育、交通、工业制造、文化娱乐等行业都有非常广泛的应用场景。只要是有防篡改数据记录、审计需求，业务上涉及交易、结算、清算、仲裁的行业，都是区块链 + 的潜在应用对象。

11.3 未来展望

中国古代六经之首的《易经》强调“象、数、理、占”。“象”可以简单理解为现象；“数”就是涉及现象中有关计算的数据属性；而“理”就是隐含在现象和数据中的规律、道理；“占”实际上就是计算，特别是带有预测性质的计算。古希腊毕达哥拉斯学派认为，数是万物的本原，事物的性质是由某种数量关系决定的，万物按照一定的数量比例而构成和谐的秩序。毕达哥拉斯学派的观点对后来的柏拉图、甚至文艺复兴时期的思想都有极其重要的影响。由此看出，一部计算的发展史，贯穿了人类的文明发展史。

到今天，人类文明的发展到了一个前所未有的新阶段。一方面，数据量正以指数型增长速度膨胀，现有的以冯·诺依曼架构体系为基础的 IT 架构似乎已经接近其能力的极限。长远来说，地球上数据的存储介质也存在极限。因此，人类走出地球，奔向宇宙的驱动力可能不仅仅是来自居住空间需求的驱动，更大的驱动力可能来自数据增长的驱动。另一方面，我们也看到，以图灵机模型为基础发展起来的电脑科技，已经使人类看到了未来无机物智能超越人类智能的可能。这两方面都给人类文明带来极大挑战。可以说，人类已经别无选择，未来只能靠非生物计算来应对数据膨胀带来的挑战，这就需要有革命性的新计算技术。而在这一方面，非冯·诺依曼架构的神经系统芯片、量子计算机已经逐渐走出实验室，给人们的希望带来了一线曙光。

但人类也更担心由非生物计算发展起来的日益强大的非生物智能会对人类伦理、社会，甚至生存产生很大的威胁。如何解决这两方面的矛盾，是一个值得整个人类思考的问题。根据过往的历史，我们不妨大胆假设，未来能真正解决该问题的，一定不是某一项技术，而是集多种技术为一体的某种新架构。我们也可以更进一步地大胆假设，在这个新架构中，量子计算可以解决计算能力的问题；神经系统计算可以解决智能认知的问题；而更关键的是，区块链可以解决电脑、机器人行为规范、自治管理的问题。

如此想来，未来还是充满机遇，因此我们也对未来充满憧憬和希望。

推荐阅读

数据挖掘：实用案例分析

作者：张良均 等 ISBN：978-7-111-42591-5 定价：79.00元

MATLAB数据分析与挖掘实战

作者：张良均 等 ISBN：978-7-111-50435-1 定价：69.00元

R语言数据分析与挖掘实战

作者：张良均 等 ISBN：978-7-111-51604-0 定价：69.00元

Python数据分析与挖掘实战

作者：张良均 等 ISBN：978-7-111-52123-5 定价：69.00元

Hadoop大数据分析与挖掘实战

作者：张良均 等 ISBN：978-7-111-52265-2 定价：69.00元

推荐阅读

云计算：概念、技术与架构

作者：Thomas Erl 等 ISBN：978-7-111-46134-0 定价：69.00元

企业应用架构模式

作者：Martin Fowler ISBN：978-7-111-30393-0 定价：59.00元

设计模式：可复用面向对象软件的基础

作者：Erich Gamma 等 ISBN：7-111-07575-2 定价：35.00元

深入理解云计算：基本原理和应用程序编程技术

作者：拉库马·布亚 等 ISBN：978-7-111-49658-8 定价：69.00元

云计算与分布式系统：从并行处理到物联网

作者：Kai Hwang 等 ISBN：978-7-111-41065-2 定价：85.00元

作者简介

邹　均

中关村区块链产业联盟专家、服务合约（Service Contract）方向博士，关注与实践区块链技术与应用，现为海纳云CTO。曾任IBM澳洲金融行业首席软件架构师。擅长云计算、大数据、软件定义存储。融智北京高端外国专家，在国际会议期刊发表论文20余篇。

张海宁

VMware中国云原生应用首席架构师，Harbor企业级开源容器Registry项目负责人，Cloud Foundry中国社区最早的技术布道师之一，多年软件开发经验。曾任IBM资深软件工程师、Sun公司资深架构师等。目前着重关注容器、云计算和区块链领域的研究和开发。

唐　屹

广州大学教授、理学博士，专注于网络信息安全、分布式计算、区块链安全及应用等，为国外知名安全公司开发过椭圆曲线密码软件，获密码科技进步二等奖（省部级）。多次主持或参与完成国家级科技与人才项目基金工作。

李　磊

合肥工业大学副教授，Macquarie大学博士。擅长数据挖掘、社会计算、智能计算。多次担任IEEE国际会议程序委员会委员与组织者，在社会计算和区块链等领域发表论文40余篇，被引用350余次。

封面设计：李雪菲

现在市面上有关区块链的书都是在讲解区块链的概念及应用场景，描述区块链技术的书却很少。我们希望读者能多了解区块链技术，多发展区块链技术，并且加以应用。只有我们了解区块链技术之后，才能真正理解区块链的意义，而不会随波逐流，人云亦云，并且有自己的判断，希望读者们能够认真读这本书，了解区块链技术，相信必定会大有所获。

——蔡维德

美国亚利桑那州立大学荣誉教授，北航区块链实验室主任

未来已来，只是尚未流行。围绕区块链技术的未来全球共享式经济协作，已经悄然来到我们身边。但是要让这种经济模式普及开来，还需要更多人去理解和使用这项技术，不仅仅是了解其简单的概念和空泛的口号，而是需要去切切实实理解其深层运作机制，这样才能更好地接受或改进这项技术。本书作为国内第一本介绍区块链技术方面的书籍，为读者进行区块链实践展示了一个全景图，是区块链学习、研究、开发的好帮手。

——张斌

联动优势科技有限公司CEO

在未来，区块链所联结的，不会像比特币一样是无法辨别的匿名账户和价值不定的虚拟资产，而是千千万万真实存在的个体和公司实体，上面所承载的资产，都将具有现实的价值和对应物，而这个虚拟的网络上发生的一切，也都会直接作用于现实世界。

——邓迪

太一云科技有限公司董事长兼CEO

这次邹均先生主编的这本区块链的书，相信一定会在IT业内，特别是在企业IT架构圈内产生巨大的反响，一定会深受广大区块链爱好者、参与者、实践者的热烈欢迎。

——黎江

北京世纪互联创新研究院院长

上架指导：计算机\数据库

ISBN 978-7-111-55356-4

9 787111 553564 >

定价：69.00元

投稿热线：(010) 88379604
客服热线：(010) 88379426 88361066
购书热线：(010) 68326294 88379649 68995259

华章网站：www.hzbook.com
网上购书：www.china-pub.com
数字阅读：www.hzmedia.com.cn